



Hiroyuki Waki et al.  
S.N. 09/288,263

日本国特許庁

PATENT OFFICE  
JAPANESE GOVERNMENT

NAKL-BG55

別紙添付の書類に記載されている事項は下記の出願書類に記載されて  
いる事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed  
with this Office.

出願年月日

Date of Application:

1998年 4月 8日

出願番号

Application Number:

平成10年特許願第096204号

出願人

Applicant (s):

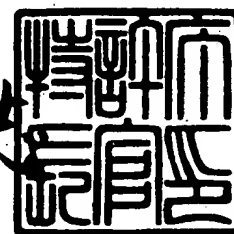
松下電器産業株式会社

CERTIFIED COPY OF  
PRIORITY DOCUMENT

1999年 3月 5日

特許庁長官  
Commissioner,  
Patent Office

山田佐平



出証番号 出証特平11-3011851

【書類名】 特許願

【整理番号】 2022590401

【提出日】 平成10年 4月 8日

【あて先】 特許庁長官 荒 井 寿 光 殿

【国際特許分類】 G06F 5/00

【発明の名称】 仮想マシン及びコンパイラ

【請求項の数】 30

【発明者】

    【住所又は居所】 大阪府門真市大字門真1006番地 松下電器産業株式会社内

    【氏名】 和氣 裕之

【発明者】

    【住所又は居所】 大阪府門真市大字門真1006番地 松下電器産業株式会社内

    【氏名】 井上 信治

【発明者】

    【住所又は居所】 大阪府門真市大字門真1006番地 松下電器産業株式会社内

    【氏名】 葉山 悟

【発明者】

    【住所又は居所】 大阪府門真市大字門真1006番地 松下電器産業株式会社内

    【氏名】 藤田 光子

【発明者】

    【住所又は居所】 大阪府門真市大字門真1006番地 松下電器産業株式会社内

    【氏名】 石川 亮

【特許出願人】

    【識別番号】 000005821

【氏名又は名称】 松下電器産業株式会社

【代理人】

【識別番号】 100090446

【弁理士】

【氏名又は名称】 中島 司朗

【手数料の表示】

【予納台帳番号】 014823

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 9003742

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 仮想マシン及びコンパイラ

【特許請求の範囲】

【請求項 1】

実マシンによる制御の下で仮想マシン命令を実行するスタック型仮想マシンであって、

データを後入れ先出し方式で一時的に記憶するためのスタック手段と、

実行対象となる仮想マシン命令列と、それら仮想マシン命令それぞれに対応づけられた情報であって、対応づけられた仮想マシン命令に後続する仮想マシン命令が実行された場合の前記スタック手段でのデータの格納状態の変化を示す先行命令情報とを記憶する命令記憶手段と、

前記命令記憶手段から次に実行すべき仮想マシン命令及び対応する先行命令情報を読み出す読み出し手段と、

読み出された仮想マシン命令と先行命令情報との組み合わせに対応する演算処理を特定し実行する解読実行手段と

を備えることを特徴とする仮想マシン。

【請求項 2】

前記解読実行手段は、

対象とする全ての種類の仮想マシン命令と全ての種類の先行命令情報との組み合わせそれぞれに対応する実マシン命令列を記憶する実マシン命令列記憶部と、

前記読み出し手段によって読み出された仮想マシン命令と先行命令情報との組み合わせに対応する実マシン命令列を前記実マシン命令列記憶部に記憶された実マシン命令列から特定する特定部と、

特定された実マシン命令列を実行する実行部と

を有することを特徴とする請求項 1 記載の仮想マシン。

【請求項 3】

前記先行命令情報は、対応する仮想マシン命令に後続する仮想マシン命令が実行された場合の前記スタック手段に格納されるデータ数の増減を示し、

前記実マシン命令列記憶部に記憶された実マシン命令列には、対応する先行命



令情報に基づいて前記スタック手段を先行処理する実マシン命令が含まれていることを特徴とする請求項 2 記載の仮想マシン。

【請求項 4】

前記実マシン命令列記憶部に記憶された実マシン命令列は、前記スタック手段において最後に格納されたデータと最後から 2 番目に格納されたデータそれぞれを記憶する領域が前記実マシンが備える 2 つのレジスタに割り当てられていることを前提とする内容であることを特徴とする請求項 3 記載の仮想マシン。

【請求項 5】

前記命令記憶手段は、前記仮想マシン命令列を記憶する第 1 領域と、その第 1 領域における各仮想マシン命令の記憶位置に関連付けられた記憶位置に前記先行命令情報を記憶する第 2 領域とからなり、

前記読み出し手段は、前記第 1 領域と前記第 2 領域それぞれの対応する記憶領域から前記仮想マシン命令と先行命令情報との組を読み出すことを特徴とする請求項 1 記載の仮想マシン。

【請求項 6】

前記命令記憶手段は、前記仮想マシン命令列と前記先行命令情報とを、対応する 1 つの仮想マシン命令と 1 つの先行命令情報との組を 1 つの拡張仮想マシン命令とする拡張仮想マシン命令列として記憶し、

前記読み出し手段は、前記命令記憶手段から拡張仮想マシン命令を読み出し、

前記解読実行手段は、読み出された拡張仮想マシン命令に対応する演算処理を特定し実行することを特徴とする請求項 1 記載の仮想マシン。

【請求項 7】

スタック型仮想マシンを対象とするコンパイラであって、

ソースプログラムを前記仮想マシンが実行する仮想マシン命令列に変換する命令列変換手段と、

変換によって得られた仮想マシン命令列を構成する仮想マシン命令それぞれについて、後続する仮想マシン命令が前記仮想マシンによって実行された場合の前記スタック手段でのデータの格納状態の変化を示す先行命令情報を生成する先行命令情報生成手段と、

生成された先行命令情報と対応する仮想マシン命令とを関連付けて出力する関連付け手段と

を備えることを特徴とする仮想マシンコンパイラ。

【請求項 8】

実マシンによる制御の下で仮想マシン命令を実行する仮想マシンであって、

実行対象となる仮想マシン命令列を記憶する命令記憶手段と、

前記命令記憶手段から次に実行すべき仮想マシン命令を読み出す読み出し手段と、

読み出された仮想マシン命令に対応する演算処理を特定し実行する解読実行手段とを備え、

前記解読実行手段は、

読み出された仮想マシン命令が実行制御の流れを分岐させる分岐命令であるか否かを判定する分岐命令判定部と、

分岐命令であると判定される度に、その分岐命令の実行に加えて、この仮想マシンへの割込み要求の発生の有無の検出と、発生している場合の割込み処理とを実行する割込み処理部とを有することを特徴とする仮想マシン。

【請求項 9】

前記解読実行手段はさらに、

対象とする全ての種類の仮想マシン命令に対応する実マシン命令列とこの仮想マシンへの割込み要求に対処するための実マシン命令列を記憶する実マシン命令列記憶部と、

指定された実マシン命令列を実行する実行部とを備え、

前記割込み処理部は、前記分岐命令判定部によって分岐命令であると判定される度に、前記実行部に、前記割込み要求に対処するための実マシン命令列を実行させた後に前記分岐命令に対応する実マシン命令列を実行させることを特徴とする請求項 8 記載の仮想マシン。

【請求項 10】

実マシンによる制御の下で仮想マシン命令を実行する仮想マシンであって、

実行対象となる仮想マシン命令列を記憶する命令記憶手段と、

前記命令記憶手段から次に実行すべき仮想マシン命令を読み出す読み出し手段と、

読み出された仮想マシン命令に対応する演算処理を特定し実行する解読実行手段とを備え、

前記解読実行手段は、

読み出された仮想マシン命令が一定個数の仮想マシン命令の集まりをブロックとした場合の各ブロックを代表する仮想マシン命令に相当するか否かを判定するブロック判定部を備え、

各ブロックを代表する仮想マシン命令であると判定される度に、その仮想マシン命令の実行に加えて、この仮想マシンへの割込み要求の発生の有無の検出と、発生している場合の割込み処理とを実行する割込み処理部とを有することを特徴とする仮想マシン。

#### 【請求項 11】

前記解読実行手段はさらに、

対象とする全ての種類の仮想マシン命令に対応する実マシン命令列とこの仮想マシンへの割込み要求に対処するための実マシン命令列を記憶する実マシン命令列記憶部と、

指定された実マシン命令列を実行する実行部とを備え、

前記ブロック判定部は、仮想マシン命令が読み出される度に、それまでに読み出された仮想マシン命令の総数が一定値の倍数に一致するか否かを比較し、一致する場合に、前記仮想マシン命令は各ブロックを代表する仮想マシン命令に相当すると判定し、

前記割込み処理部は、前記分岐命令判定部によって各ブロックを代表する仮想マシン命令であると判定される度に、前記実行部に、前記割込み要求に対処するための実マシン命令列を実行させた後に前記仮想マシン命令に対応する実マシン命令列を実行させることを特徴とする請求項 10 記載の仮想マシン。

#### 【請求項 12】

実マシンによる制御の下で仮想マシン命令を実行する仮想マシンであって、

実マシン命令からなる複数のサブプログラムを記憶する実マシンプログラム記

憶手段と、

実行対象となる仮想マシン命令列を記憶する第1領域と、前記実マシンプログラム記憶手段における各サブプログラムへのポインタを記憶する第2領域とを含む命令記憶手段と、

前記命令記憶手段の第1領域から次に実行すべき仮想マシン命令を読み出す読み出し手段と、

読み出された仮想マシン命令に対応する演算処理を特定し実行する解読実行手段とを備え、

前記解読実行手段は、

読み出された仮想マシン命令が前記第2領域に実行制御を移す命令であるか否かを判定する領域判定部と、

前記第2領域に実行制御を移す命令であると判定された場合に、その移動先の第2領域の箇所に格納された前記ポインタが示す前記サブプログラムを実行するアドレス変換実行部と

を有することを特徴とする仮想マシン。

#### 【請求項13】

前記命令記憶手段における第1領域と第2領域とは、一定のアドレス値を境界とする隣接した記憶位置に設けられ、

前記領域判定部は、読み出された仮想マシン命令がサブプログラムを呼び出す命令である場合に、その呼び出し先アドレスと前記境界との大小関係を比較することによって前記判定を行うことを特徴とする請求項12記載の仮想マシン。

#### 【請求項14】

実マシンによる制御の下で仮想マシン命令を実行する仮想マシンであって、

実行対象となる仮想マシン命令列を記憶する命令記憶手段と、

前記命令記憶手段から次に実行すべき仮想マシン命令を読み出す読み出し手段と、

読み出された仮想マシン命令に対応する演算処理を特定し実行する解読実行手段とを備え、

前記命令記憶手段は、前記仮想マシン命令列を構成する基本ブロックそれぞれ

に対応する命令ブロックの集まりからなり、

各命令ブロックは、その命令ブロックの前記命令記憶手段における先頭位置を特定するための識別子を格納した識別子領域と、対応する基本ブロックの非分岐命令だけを格納した非分岐命令領域と、対応する基本ブロックの分岐命令を格納した分岐命令領域とを含み、

前記分岐命令領域に格納された分岐命令は、その分岐先が前記識別子によって指定され、

前記解読実行手段は、読み出された仮想マシン命令が前記分岐命令である場合には、その分岐先として指定されている識別子に対応する命令ブロックの非分岐命令領域の先頭に実行制御を分岐させることを特徴とする仮想マシン。

#### 【請求項 15】

前記解読実行手段は、

次に読み出すべき仮想マシン命令が属する命令ブロックの識別子を格納する識別子レジスタと、その仮想マシン命令のその命令ブロックにおける相対的な記憶位置を示すオフセットを格納するオフセットカウンタとからなるプログラムカウンタを備え、

前記読み出し手段は、前記プログラムカウンタに格納された識別子とオフセットに基づいて前記仮想マシン命令を読み出し、

前記解読実行手段は、読み出された仮想マシン命令が前記分岐命令である場合には、その分岐先として指定されている識別子を前記識別子レジスタに書き込むと共に前記オフセットカウンタを初期値にリセットし、読み出された仮想マシン命令が前記非分岐命令である場合には、前記オフセットカウンタを増加させ、

前記読み出し手段は、前記解読実行手段によって更新されたプログラムカウンタの識別子とオフセットに基づいて次に実行すべき仮想マシン命令を読み出すことを特徴とする請求項 14 記載の仮想マシン。

#### 【請求項 16】

前記解読実行手段はさらに、対象とする全ての種類の仮想マシン命令それぞれに対応する実マシン命令列を記憶する実マシン命令列記憶部を備え、

前記命令記憶手段の各命令ブロックはさらに、その命令ブロックの非分岐命令

領域及び分岐命令領域に格納された仮想マシン命令それぞれに対応する実マシン命令列を前記実マシン命令列記憶部に記憶された実マシン命令列から特定するためのデコードデータを含むデコードデータ列を格納したデコードデータ列領域を含み、

前記読み出し手段は、デコードデータ列が格納された命令ブロックについては前記仮想マシン命令に代えて前記デコードデータを読み出し、デコードデータ列が格納されていない命令ブロックについては仮想マシン命令を読み出した後に、その仮想マシン命令に対応する実マシン命令列を前記実マシン命令列記憶部に記憶された実マシン命令列から特定するためのデコードデータを生成し、

前記解読実行手段は、前記読み出し手段から読み出し又は生成されたデコードデータによって特定される実マシン命令列を前記実マシン命令列記憶部から読み出して実行することを特徴とする請求項 15 記載の仮想マシン。

#### 【請求項 17】

前記命令記憶手段の各命令ブロックはさらに、その命令ブロックのデコードデータ列領域に前記デコードデータが格納されているか否かを示すフラグを格納したフラグ領域を含み、

前記解読実行手段はさらに、分岐命令を実行した場合には、その分岐先となる命令ブロックのフラグ領域に格納されたフラグを読み出して保持するカレントフラグ記憶部を備え、

前記読み出し手段は、前記カレントフラグ記憶部に保持されたフラグに基づいて、デコードデータの読み出し又は仮想マシン命令の読み出しを行うことを特徴とする請求項 16 記載の仮想マシン。

#### 【請求項 18】

前記命令記憶手段の各命令ブロックはさらに、その命令ブロックのデコードデータ列領域に前記デコードデータが格納されているか否かを示すフラグを格納したフラグ領域を含み、

前記解読実行手段はさらに、分岐命令を実行した場合にその分岐先となる命令ブロックのフラグ領域に格納されたフラグを参照することにより、その命令ブロックにはデコードデータが格納されていないと判定したときには、その命令プロ

ックに格納されている仮想マシン命令列を読み出して対応するデコードデータ列に変換し、その命令ブロックのデコードデータ列領域に書き込むデコードデータ列書き込み部を備えることを特徴とする請求項 16 記載の仮想マシン。

【請求項 19】

実マシンによる制御の下で仮想マシン命令を実行する仮想マシンであって、  
 実行対象となる仮想マシン命令列を圧縮符号で記憶する命令記憶手段と、  
 前記命令記憶手段から次に実行すべき仮想マシン命令の圧縮符号を読み出し、  
 対応する仮想マシン命令に復号する読み出し手段と、  
 復号された仮想マシン命令に対応する演算処理を特定し実行する解読実行手段  
 とを備え、

前記命令記憶手段は、前記仮想マシン命令列を構成する基本ブロックそれぞれ  
 に対応する命令ブロックの集まりからなり、

各命令ブロックは、その命令ブロックの前記命令記憶手段における先頭位置を  
 特定するための識別子を格納した識別子領域と、対応する基本ブロックの非分岐  
 命令だけを格納した非分岐命令領域と、対応する基本ブロックの分岐命令を格納  
 した分岐命令領域とを含み、

前記分岐命令領域に格納された分岐命令は、その分岐先が前記識別子によって  
 指定され、

前記解読実行手段は、復号された仮想マシン命令が前記分岐命令である場合に  
 は、その分岐先として指定されている識別子に対応する命令ブロックの非分岐命  
 令領域の先頭に実行制御を分岐させることを特徴とする仮想マシン。

【請求項 20】

前記命令記憶手段の各命令ブロックにはさらに、その命令ブロックに格納され  
 た仮想マシン命令の圧縮符号を伸長復号するための情報テーブルであって、対応  
 する圧縮符号と仮想マシン命令との組の集まりからなる復号テーブルを格納した  
 復号テーブル領域を含み、

前記読み出し手段は、次に実行すべき仮想マシン命令が属する命令ブロックの  
 復号テーブルを参照しながら、前記命令記憶手段から読み出した圧縮符号を対応  
 する仮想マシン命令に復号することを特徴とする請求項 19 記載の仮想マシン。

## 【請求項 21】

前記解読実行手段は、

次に読み出すべき仮想マシン命令が属する命令ブロックの識別子を格納する識別子レジスタと、その仮想マシン命令のその命令ブロックにおける相対的な記憶位置を示すオフセットを格納するオフセットカウンタとからなるプログラムカウンタを備え、

前記読み出し手段は、前記プログラムカウンタに格納された識別子とオフセットに基づいて前記仮想マシン命令を読み出し、

前記解読実行手段は、読み出された仮想マシン命令が前記分岐命令である場合には、その分岐先として指定されている識別子を前記識別子レジスタに書き込むと共に前記オフセットカウンタを初期値にリセットし、読み出された仮想マシン命令が前記非分岐命令である場合には、前記オフセットカウンタを増加させ、

前記読み出し手段は、前記解読実行手段によって更新されたプログラムカウンタの識別子とオフセットに基づいて次に実行すべき仮想マシン命令を読み出すことを特徴とする請求項 20 記載の仮想マシン。

## 【請求項 22】

前記解読実行手段はさらに、対象とする全ての種類の仮想マシン命令それぞれに対応する実マシン命令列を記憶する実マシン命令列記憶部を備え、

前記命令記憶手段の各命令ブロックはさらに、その命令ブロックの非分岐命令領域及び分岐命令領域に格納された仮想マシン命令それぞれに対応する実マシン命令列を前記実マシン命令列記憶部に記憶された実マシン命令列から特定するためのデコードデータを含むデコードデータ列を格納したデコードデータ列領域を含み、

前記読み出し手段は、デコードデータ列が格納された命令ブロックについては前記仮想マシン命令に代えて前記デコードデータを読み出し、デコードデータ列が格納されていない命令ブロックについては仮想マシン命令の圧縮符号を読み出して対応する仮想マシン命令に復号した後に、その仮想マシン命令に対応する実マシン命令列を前記実マシン命令列記憶部に記憶された実マシン命令列から特定するためのデコードデータを生成し、



前記解説実行手段は、前記読み出し手段から読み出し又は生成されたデコードデータによって特定される実マシン命令列を前記実マシン命令列記憶部から読み出して実行することを特徴とする請求項 21 記載の仮想マシン。

【請求項 23】

前記命令記憶手段の各命令ブロックはさらに、その命令ブロックのデコードデータ列領域に前記デコードデータが格納されているか否かを示すフラグを格納したフラグ領域を含み、

前記解説実行手段はさらに、分岐命令を実行した場合には、その分岐先となる命令ブロックのフラグ領域に格納されたフラグを読み出して保持するカレントフラグ記憶部を備え、

前記読み出し手段は、前記カレントフラグ記憶部に保持されたフラグに基づいて、デコードデータの読み出し又は仮想マシン命令の読み出しを行うことを特徴とする請求項 22 記載の仮想マシン。

【請求項 24】

前記命令記憶手段の各命令ブロックはさらに、その命令ブロックのデコードデータ列領域に前記デコードデータが格納されているか否かを示すフラグを格納したフラグ領域を含み、

前記解説実行手段はさらに、分岐命令を実行した場合にその分岐先となる命令ブロックのフラグ領域に格納されたフラグを参照することにより、その命令ブロックにはデコードデータが格納されていないと判定したときには、その命令ブロックに格納されている仮想マシン命令列の圧縮符号を読み出して対応する仮想マシン命令列に復号した後に、対応するデコードデータ列に変換し、その命令ブロックのデコードデータ列領域に書き込むデコードデータ列書込み部を備えることを特徴とする請求項 22 記載の仮想マシン。

【請求項 25】

実マシンによる制御の下で仮想マシン命令を実行する仮想マシンと共に用いられ、実行対象となる一部の仮想マシン命令列をその実行に先立って実マシン命令列に変換するその場式コンパイラであって、

前記仮想マシン命令列を構成する仮想マシン命令それぞれについて、前記仮想

マシン命令列を基本ブロックに分割した場合の各基本ブロックの先頭となる仮想マシン命令であるか否かを示すブロック先頭情報の入力を受け付けるブロック先頭情報獲得手段と、

前記仮想マシン命令列を構成する仮想マシン命令それぞれについて、対応する実マシン命令列に変換する変換手段と、

前記変換手段によって得られた実マシン命令列における実マシン命令であって、前記ブロック先頭情報獲得手段が受け付けたブロック先頭情報によって特定された基本ブロックの先頭となる仮想マシン命令に対応する実マシン命令が、前記実マシンによってアドレッシングされ得ない位置に配置される命令であるか否かを判定する分岐違反判定手段と、

前記分岐違反判定手段によって前記実マシン命令が前記位置に配置される命令であると判定された場合に、前記実マシン命令が前記位置に配置されないよう1個以上の無動作命令を前記実マシン命令列に追加挿入して出力する出力手段とを備えることを特徴とするコンパイラ。

#### 【請求項 26】

前記出力手段は、前記実マシン命令が前記実マシンによってアドレッシングされ得る後続位置にずらせるために必要な個数の無動作命令を前記実マシン命令列における基本ブロックの先頭位置に追加挿入することを特徴とする請求項 25 記載のコンパイラ。

#### 【請求項 27】

実マシンによる制御の下で仮想マシン命令を実行するスタック型仮想マシンにおいて、実行対象となる仮想マシン命令列を格納しておくための命令記憶手段における前記格納方法であって、

実行対象となる仮想マシン命令列と、それら仮想マシン命令それぞれに対応づけられた情報であって、対応づけられた仮想マシン命令に後続する仮想マシン命令が実行された場合の前記スタックでのデータの格納状態の変化を示す先行命令情報とを関連付けて格納することを特徴とする仮想マシン命令列の格納方法。

#### 【請求項 28】

コンピュータをスタック型仮想マシンとして機能させるためのプログラムを記

録した記録媒体であって、

前記仮想マシンは、

データを後入れ先出し方式で一時的に記憶するためのスタック手段と、

実行対象となる仮想マシン命令列と、それら仮想マシン命令それぞれに対応づけられた情報であって、対応づけられた仮想マシン命令に後続する仮想マシン命令が実行された場合の前記スタック手段でのデータの格納状態の変化を示す先行命令情報とを記憶する命令記憶手段と、

前記命令記憶手段から次に実行すべき仮想マシン命令及び対応する先行命令情報を読み出す読み出し手段と、

読み出された仮想マシン命令と先行命令情報との組み合わせに対応する演算処理を特定し実行する解釈実行手段と

を備えることを特徴とする記録媒体。

#### 【請求項 29】

実マシンによる制御の下で仮想マシン命令を実行する仮想マシンにおいて、実行対象となる仮想マシン命令列を格納しておくための命令記憶手段における前記格納方法であって、

前記命令記憶手段は、前記仮想マシン命令列を構成する基本ブロックそれぞれに対応する命令ブロックの集まりからなり、

各命令ブロックは、その命令ブロックの前記命令記憶手段における先頭位置を特定するための識別子を格納した識別子領域と、対応する基本ブロックの非分岐命令だけを格納した非分岐命令領域と、対応する基本ブロックの分岐命令を格納した分岐命令領域とを含み、

前記分岐命令領域に格納された分岐命令は、その分岐先が前記識別子によって指定されていることを特徴とする仮想マシン命令列の格納方法。

#### 【請求項 30】

コンピュータを仮想マシンとして機能させるためのプログラムを記録した記録媒体であって、

前記仮想マシンは、

実行対象となる仮想マシン命令列を記憶する命令記憶手段と、

前記命令記憶手段から次に実行すべき仮想マシン命令を読み出す読み出し手段と、

読み出された仮想マシン命令に対応する演算処理を特定し実行する解読実行手段とを備え、

前記命令記憶手段は、前記仮想マシン命令列を構成する基本ブロックそれぞれに対応する命令ブロックの集まりからなり、

各命令ブロックは、その命令ブロックの前記命令記憶手段における先頭位置を特定するための識別子を格納した識別子領域と、対応する基本ブロックの非分岐命令だけを格納した非分岐命令領域と、対応する基本ブロックの分岐命令を格納した分岐命令領域とを含み、

前記分岐命令領域に格納された分岐命令は、その分岐先が前記識別子によって指定され、

前記解読実行手段は、読み出された仮想マシン命令が前記分岐命令である場合には、その分岐先として指定されている識別子に対応する命令ブロックの非分岐命令領域の先頭に実行制御を分岐させることを特徴とする記録媒体。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本発明は、仮想マシン及び仮想マシンコンパイラに関し、特に仮想マシンの実行速度を向上させる技術に関する。

【0002】

【従来の技術】

（仮想マシンの一般技術）

CPU（以下、「実マシン」と呼ぶ。）の種別に依存することなく、同一のプログラムをそのまま、各種CPUを備えるパーソナルコンピュータやワークステーション等のコンピュータ上で実行させる技術として仮想マシンがある。情報通信分野、特に、異機種のコピュータ同士が接続されるネットワークにおいては、共有資源の迅速かつ円滑な利用のために、このような実マシンのアーキテクチャの差異を吸収する技術である仮想マシンの果たす役割は大きい。

## 【0003】

仮想マシンは、仮想マシン固有の命令（以下、「仮想マシン命令」と呼ぶ。）の集まりからなる実行形式のプログラム（以下、「仮想マシンプログラム」又は「仮想マシン命令列」と呼ぶ。）をソフトウェア的に解釈・実行する仮想的なプロセッサであり、一般に、仮想マシン自体は、実マシン用の命令（以下、「実マシン命令」と呼ぶ。）で書かれたプログラム（以下、「実マシンプログラム」又は「実マシン命令列」と呼ぶ。）で実現されている。そして、仮想マシンの中心的な課題である実行速度の確保という点から、仮想マシン自体のアーキテクチャとしてはスタック型（スタックマシン）が採られる。

## 【0004】

従来のスタック型仮想マシンとして、米国 Sun Microsystems, Inc. 社が開発した Java（登録商標）がある。

図 7 1 は、Java のような従来のスタック型仮想マシン 4400 の構成を示す機能ブロック図である。従来の仮想マシン 4400 は、実行対象となる仮想マシンプログラムを保持する命令格納部 4401、その命令格納部 4401 から仮想マシン命令を読み出して解釈するデコード部 4402、その解釈結果であるデコードデータに従ってオペレーションを実行する実行部 4410 及びその実行のための一時的な後入れ先出し方式の記憶領域であるスタック 4420 から構成される。なお、本図において、実線はデータの流れ、点線は制御の流れを示す。

## 【0005】

デコード部 4402 は、さらに、この仮想マシン 4400 が実行する全ての仮想マシン命令それぞれに対応するマイクロプログラム（これは、実行部 4410 に格納されている。）へのジャンプアドレス等を予め記憶するデコードテーブル 4406、命令格納部 4401 から次に読み出すべき命令のアドレスを保持するプログラムカウンタ（PC）4404、その命令を読み出す読み込み部 4403 及び読み出した命令に対してデコードテーブル 4406 を参照することにより対応するジャンプアドレスを特定して実行部 4410 に出力する検索部 4405 を有する。ここで、マイクロプログラムとは、個々の仮想マシン命令のオペレーションに対応する実マシンプログラムを意味する。

## 【0006】

実行部4410は、上記マイクロプログラム、つまり、全ての仮想マシン命令それぞれに対応する実マシンプログラムを予め上記ジャンプアドレスで始まる箇所に記憶しているマイクロプログラム記憶部4411及びスタック4420の最上位アドレスを示すスタックポインタ（SP）4412を有する。

図72は、この仮想マシン4400の命令セットを説明するための図である。本図には、この仮想マシン4400が解釈・実行する全ての仮想マシン命令の二モニク表示、オペレーションの意味、スタックの状態変化及び実行後のスタックポインタの変化が示されている。本図において、表記“s0”はスタックの最上位の値を表し、“s1”はスタックの最上位から2番目の値を表す。例えば仮想マシン命令“Add”の表記“s0←s0+s1”は、実行後にスタックの最上位の値が、実行前のスタックの最上位の値と、スタックの最上位から2番目の値を加えたものとなることを意味する。表記“sp←sp-1”は、実行後にスタックの階層が一つ減ることを意味する。

## 【0007】

図73は、図71に示されたデコードテーブル4406の内容を示す図である。各仮想マシン命令の種別を示すオペコード4406aと、その仮想マシン命令に対応するマイクロプログラム記憶部4411内のマイクロプログラムへのジャンプアドレス4406bと、その仮想マシン命令に含まれるオペランド数4406cとが格納されている。なお、全てのオペコードは1バイト長とし、オペランドの単位も1バイトとする。また、物理的なビット配列で表現された仮想マシン命令（オペコードだけやオペランドだけを指すこともある）を「仮想マシンコード」と呼ぶ。

## 【0008】

図74（a）～（d）は、それぞれ、図71に示されたマイクロプログラム記憶部4411に格納されているマイクロプログラムのうち、仮想マシン命令“Push”、“Add”、“Mult”に対応するもの及びそれらに共通する後半部分（次の仮想マシン命令にジャンプするための実マシンプログラム）のリストを示す。ここで用いられている実マシン命令の意味は、図75に示される通りである。つまり、こ

の仮想マシン4400自体は、図75に示される実マシン命令を解釈・実行する実マシンによって実行されるものとしている。なお、物理的には、仮想マシン4400のPC4404は実マシンのレジスタ2番(r2)に、仮想マシン4400のSP4412は実マシンのレジスタ3番(r3)に割り当てられている。

#### 【0009】

図76は、図71に示されたデコード部4402の動作を示すフローチャートである。読み込み部4403は、実行部4410から信号線Rを介して「次の命令を読み込む」旨の指示を受けると(ステップ4502~4503)、PC4404に格納されたアドレスに基づいて命令格納部4401に格納された仮想マシン命令を読み出す(ステップ4504~4505)。続いて、検索部4405は、デコードテーブル4406を参照することにより、その仮想マシン命令に対応するジャンプアドレスと必要なオペランドとをデコードデータとして実行部4410に出力すると共に(ステップ4506)、1個の仮想マシン命令に対する解釈の区切りを示す旨の通知「読み込み終了」を信号線Rを介して実行部4410に通知する(ステップ4507)。

#### 【0010】

図77は、図76のフローチャートにおけるステップ4506の詳細を示す図である。検索部4405は、まず、読み込み部4403によって読み込まれた1バイト分の仮想マシンコード(オペコード)とデコードテーブル4406中のオペコード4406aとを逐一に比較し(ステップ4802~4807)、一致した場合には、対応するジャンプアドレス4406bとオペランド数4406cとをデコードテーブル4406から読み出す。そして、そのジャンプアドレス4406bを実行部4410に出力した後に(ステップ4808)、そのオペランド数4406cに相当するオペランドを読み込み部4403を介して命令格納部4401から読み出し、実行部4410に出力する(ステップ4809~4813)。

#### 【0011】

なお、図76及び図77に示されたフローチャートは、デコード部4402から出力されたデコードデータが直接実行部4410に渡されることを前提とするものであるが、複数のデコードデータを記憶できるバッファを介して実行部44

10に渡されることを前提とした場合には、デコード部4402の動作は、図78に示されるフローチャートの通りとなる。つまり、バッファの記憶容量が許す限り、実行部4410による実行とは独立して、命令格納部4401からの仮想マシン命令の読み出しと解釈とを行う（ステップ4605～4613）。

#### 【0012】

図79は、図71に示された実行部4410の動作を示すフローチャートである。実行部4410は、SP4412とPC4404とを初期化しておいた後に（ステップ4702）、個々の仮想マシン命令ごとに以下の動作を繰り返す（ステップ4703～4707）。つまり、読み込み部4403に対して信号線Rを介して「次の命令を読み込む」旨の指示を通知した後に（ステップ4703）、その信号線Rから「読み込み終了」の通知を受けるまで、検索部4405から送られてくるデコードデータを読み込み、そのデコードデータに含まれるジャンプアドレスにジャンプすることで、その仮想マシン命令に相当するマイクロプログラム記憶部4411内のマイクロプログラムを実行する（ステップ4704～4707）。

#### 【0013】

図80（a）は、従来の仮想マシン4400の具体的な動作例を説明するためのサンプルプログラムのリストを示す。つまり、命令格納部4401には、図80（b）に示される演算式「 $2 \times (3+4)$ 」を計算する仮想マシンプログラムが置かれているとする。

図80（c）は、図80（a）に示された仮想マシンプログラムが従来の仮想マシン4400によって解釈・実行された場合において、デコード部4402から順次出力されるデコードデータを示す図である。つまり、デコード部4402は、解釈した仮想マシン命令に相当するマイクロプログラムのジャンプアドレスと必要なオペランドをデコードデータとして順次実行部4410に渡す。

#### 【0014】

図81（a）及び（b）は、それぞれ、図80（c）に示されたデコードデータ列に従って実行部4410が図80（a）の仮想マシンプログラムを実行した場合における各仮想マシン命令が実行される前後でのPC4404、SP441



2及びスタック4420の一連の変化の前半部と後半部を示す図である。ここで、PC4404は、図80(a)に示された仮想マシンプログラム中の次に実行すべき仮想マシン命令の左欄に付されたアドレスを差し、その初期値は「1」である。また、SP4412は、実行部4410が最後に格納又は読み出したスタック4420の最上位の位置を示し、その初期値はスタック4420が空であることを示す「-1」である。図81(a)及び(b)から分かるように、PC4404が9に達した段階で、計算式 $2*(3+4)$ の値が算出されている。

【0015】

以上のような仮想マシン4400の中心的な課題は実行速度の向上である。つまり、仮想マシンは、仮想マシン命令を解読する等の処理のためにオーバーヘッドを生じるので、実マシンによる実マシンプログラムの直接的な実行の場合に比較し、大幅に実行速度が劣る。そこで、仮想マシンの実行速度を向上させる従来の技術として、以下の手法が提案されている。

(第1の従来技術)

第1の従来技術は、スタックの最上位の記憶域(以下、「Top Of Stack (TOS) 変数」と呼ぶ。)をメモリ上ではなく、実マシンの特定のレジスタに割り当てておく手法である(「PLDI」ACM 1995, pp315-327 参照)。

【0016】

図82(a)～(d)は、この従来技術に基づいてマイクロプログラム記憶部を実装した場合の主な仮想マシン命令についてのマイクロプログラムのリストであり、上述の仮想マシン4400における図74に対応する。ここでは、物理的には、TOS変数は、実マシンのレジスタ0番(r0)に割り当てられている。また、PC4404及びSP4412は、図74(a)～(d)と同様に、それぞれレジスタ2番(r2)及びレジスタ3番(r3)に割り当てられている。

【0017】

図83(a)及び(b)は、図82(a)～(d)に示されたマイクロプログラムを備える仮想マシンによって図80(a)の仮想マシンプログラムが実行された場合のPC4404、SP4412、TOS変数4421及びスタック4420のうちメモリに割り当てられた領域4422(以下、仮想マシンのスタックの

うち、メモリに割り当てられた領域を「メモリスタック」と呼ぶ。)の一連の変化の前半部と後半部を示す図であり、上述の仮想マシン4400における図81(a)及び(b)に対応する。この手法によっても、PCが9に達した段階で、計算式 $2*(3+4)$ の値が算出されていることが分かる。

【0018】

図74(a)～(d)と図82(a)～(d)とを比較して分かるように、この第1の従来技術によれば、メモリへのアクセスの回数が減少される。加算“Add”や乗算“Mult”等の仮想マシン命令の実行においては、上述の仮想マシン4400によれば、各演算のためにスタック4420からの2回の読み出しと1回の書き込みの合計3回のメモリアccessが必要とされたが、この従来技術によれば、TOS変数がメモリ上ではなくレジスタに割り当てられているので、必要なメモリアccessはメモリスタック4422からの1回の読み出しだけで済む。従って、メモリアccessが減少された分だけ実行速度が向上され得る。

(第2の従来技術)

第2の従来技術は、「ネイティブコード化」と呼ばれている方法であり、仮想マシンプログラムの特定箇所を実マシン命令で記述しておき、その特定箇所については実マシンによって直接に実行させるという手法である。そのために、その特定箇所が実マシン命令で記述されている旨の識別子が用いられる。

【0019】

例えば、上述のJavaでは、仮想マシンプログラムを収めるクラスファイルのアクセスフラグ(構造体“method\_info”の要素である16ビットのフラグ“access\_flags”等)に定数名“ACC\_NATIVE”(256)を格納しておくことで、そのプログラム部分は実マシン命令で実装されていることを示す(米国 Sun Microsystems, Inc.社発行のJavaバイトコード及びJava仮想マシンの仕様書(1995年版)参照)。

【0020】

この従来技術によれば、特定箇所のプログラムについては、実マシンによって直接に実行されるので、その分だけ実行速度は向上され得る。

(第3の従来技術)

第3の従来技術は、「その場式コンパイラ(Just-In-Time Compiler)」と呼ば

れる手法で、仮想マシンプログラムの実行時において、必要に応じてその一部をコンパイル、即ち、実マシン命令に置き換える手法である（ローラ・リメイほか「Java言語入門」（株）プレントイスホール出版、1996年 参照）。この従来技術によれば、既にコンパイルが完了しているプログラム部分については、実マシンにより直接に実行されるので、その分だけ実行速度が向上され得る。

#### （第4の従来技術）

第4の従来技術は、ネットワーク上のサーバコンピュータに存在する仮想マシンプログラムを自らのコンピュータにダウンロードした後に実行するような形態において、予め仮想マシンプログラムのコードをLZ法やハフマン符号化などによって圧縮しておくことで、転送時間を削減する手法である（特開平7-121352や特開平8-263263など）。

#### 【0021】

この従来技術によれば、仮想マシンプログラムの転送時間の全体の処理時間に占める割合が大きい場合には、全体としての実行速度が向上され得る。

#### 【0022】

#### 【発明が解決しようとする課題】

しかしながら、上記従来技術には以下の問題点がある。

#### （第1の従来技術の問題点）

上記第1の従来技術、つまり、TOS変数を実マシンのレジスタに割り当てておく手法には、近年安価になってきたスーパースカラアーキテクチャの実マシン（以下、「スーパースカラマシン」と呼ぶ。）には向いていないという問題点、つまり、並列処理機能を有しない通常の実マシン（以下、「通常マシン」と呼ぶ。）で実行する場合と比較し、大きくは実行速度が向上されないという問題点がある。以下、その問題点を説明する。

#### 【0023】

まず、図84～図92を用いて、上記スーパースカラマシンや通常マシンなどのレジスタマシンが有するパイプラインの一般的な動作及びその表記法を説明する。

図84は、パイプラインの各ステージに対する省略記号を説明する図である。

ここでは、スーパースカラマシン及び通常マシンは、いずれも5個のステージからなるパイプラインを有するとする。

#### 【0024】

図85は、通常マシンのパイプラインの理想的な流れを示す図である。ここでは、4つの実マシン命令が1クロックごとに隙間なく順次に処理される様子が示されている。この状態では、パイプラインの各ステージは並列に動作するので、平均して、1クロックで1つの実マシン命令が実行されることになる。

一方、図86は、スーパースカラマシンのパイプラインの理想的な流れを示す図である。ここでは、スーパースカラマシンは、2本の独立したパイプラインを有するものとしている。この状態では、平均して、1クロックで2つの実マシン命令が実行されることになる。つまり、通常マシンに比べ、2倍の処理能力を持つ。

#### 【0025】

図87は、通常マシンにおいて、パイプラインにハザード（干渉）が生じた場合のパイプラインの流れを示す図である。ここでは、命令Bは、先行する命令Aに対して真のデータ依存関係（true dependency又はdata dependencyと呼ばれる関係）を持つ、つまり、命令Aによる演算結果を用いる命令としている。すると、命令Aによる演算結果は、メモリ参照ステージMEMが完了しないと明らかにならないため、直後の命令Bは、命令AのステージMEMの完了まで実行が一時的に保留される結果、ここにハザード（図中の記号“-”で示されるステージ）が生じる。なお、パイプライン構造を採る実マシンでは、先行する命令の実行が保留されると、後続の命令も次のステージに移ることができないため、本図に示されるように、命令Bの直後の命令Cも実行が一時的に保留される。

#### 【0026】

一方、図88は、スーパースカラマシンにおいて、パイプラインにハザードが生じた場合のパイプラインの流れを示す図である。ここでは、命令B1は、先行する命令A1及び命令A2に対して真のデータ依存関係を持つとしている。なお、本図において、命令C2の5クロック目にハザードが生じているのは、先行する命令B1と命令C1とによって実行に必要な2個のユニット（ALU）が使用さ

れているからである。

【0027】

図89及び図90は、それぞれ、図87及び図88の場合において、メモリ参照が行われてからその値が次に使用されるまでに2クロックの時間を必要とするときのパイプラインの流れを示す図である。実際の実マシンでは、一次キャッシュのメモリから、値を得るまでに2クロックの時間を要する場合が多いからである。なお、二次キャッシュ等にアクセスする場合は、より多くのクロック数を要する。

【0028】

図91及び図92は、それぞれ、通常マシン及びスーパースカラマシンにおいて、命令A及び命令A2がレジスタでジャンプ先を指定する命令である場合のパイプラインの流れを示す図である。ジャンプ先は、それぞれ、命令A及び命令A2のレジスタ参照ステージRFが終了しないと判明しないので、その間に無駄にフェッチされた後続命令Bは、次の第3クロック（記号"x"で示されるステージ）においてキャンセルされる。従って、ジャンプ先の命令C及び命令C1は、それぞれ命令A及び命令A2のレジスタ参照ステージRFが終了した後で実行される。

【0029】

次に、図93～図96を用いて、上記第1の従来技術に係るスーパースカラマシン及び通常マシンの具体的な問題点を説明する。

図93～図96は、いずれも、第1の従来技術に係る仮想マシンが図80（a）に示された仮想マシンプログラムを実行した場合の実マシンのパイプラインの流れ、より具体的には、アドレス7の仮想マシン命令"Add"のマイクロプログラム（図82（a））の後半部分（図82（d）に示されたジャンプ処理）と、続くアドレス8の仮想マシン命令"Mult"のマイクロプログラム（図82（c））の前半部分（乗算計算処理）におけるパイプラインの流れを示す。但し、図93～図96は、それぞれ、メモリ参照が行われてからその値が次に使用されるまでに1クロックの時間で済む場合における通常マシン、その場合におけるスーパースカラマシン、メモリ参照が行われてからその値が次に使用されるまでに2クロックの時間を必要とする場合における通常マシン、その場合におけるスーパースカ

ラマシンのパイプラインの流れを示す。

【0030】

これら一連のマイクロプログラム（図82（d）及び図82（a））には、重大な真のデータ依存関係が二つ存在する。つまり、仮想マシン命令"Add"のジャンプ処理（図82（d））におけるジャンプアドレスの読み込み命令"Load"とジャンプ命令"Jmp"との間に存在する第1の依存関係と、仮想マシン命令"Mult"の計算処理（図82（c））におけるスタックの2番目の変数を読み込む命令"Load"と乗算命令"Mult"との間に存在する第2の依存関係である。

【0031】

図93に示されたパイプラインでは、上記第1の依存関係は、実マシン命令"Load"と実マシン命令"Jump"間に挿入された実マシン命令"Inc"で吸収され、さらに上記第2の依存関係も、実マシン命令"Load"と実マシン命令"Mult"間に挿入された実マシン命令"Dec"で吸収されるため、一連の処理において生じるパイプラインの乱れは、実マシン命令"Jmp"の実行に起因する1個の命令のキャンセルだけとなり、処理全体は11クロックで完了する。

【0032】

図94に示されたパイプラインでは、上記第1及び第2の依存関係は吸収されないために、一連の処理において生じるパイプラインの乱れは、上記第1の依存関係に起因する第4クロックのハザードと、実マシン命令"Jmp"の実行に起因する5個の命令のキャンセルと、上記第2の依存関係に起因する第8クロックのハザードの合計3個所となり、処理全体は、図93の場合と同様に、11クロックを要することになる。

【0033】

図95に示されたパイプラインでは、図94の場合と同様に上記第1及び第2の依存関係は吸収ず、一連の処理において生じるパイプラインの乱れは、上記第1の依存関係に起因する第5クロックのハザードと、実マシン命令"Jmp"の実行に起因する1個の命令のキャンセルと、上記第2の依存関係に起因する第10クロックのハザードの合計3個所となり、処理全体は13クロックで完了する。

【0034】

図 9 6 に示されたパイプラインでは、図 9 4 の場合と同様に上記第 1 及び第 2 の依存関係は吸収ず、一連の処理において生じるパイプラインの乱れは、上記第 1 の依存関係に起因する第 4 及び第 5 クロックのハザードと、実マシン命令”Jump”の実行に起因する 7 個の命令のキャンセルと、上記第 2 の依存関係に起因する第 9 及び第 10 クロックのハザードの合計 3 個所となり、処理全体は、図 9 5 の場合と同様に、13 クロックを要することになる。

#### 【0035】

上記図 9 3 及び図 9 4 に示された一連の処理はいずれも 11 クロックを要していること、及び、上記図 9 5 及び図 9 6 に示された一連の処理はいずれも 13 クロックだけ要していることから分かるように、この従来技術によれば、通常マシンによる場合とスーパースカラマシンによる場合とで実行時間に差がなく、スーパースカラマシンによる並列処理の長所が生かされていない。

#### 【0036】

このように、この従来技術には、スーパースカラマシンによる実行効率が大幅に低下するという問題点があるが、これに加えて、エラーなどの例外処理やデバッグに不可欠な割込み処理の実現方法が提供されていないという問題点もある。

そのために、一般の実マシンと同様に、1 個の仮想マシン命令を実行する度に割込み状態の検出と処理を行うことになるが、それでは、仮想マシン命令を実行する度に新たなメモリアクセス（割込み状態を保持するメモリ上の変数をレジスタに読み込むためのデータ転送）が加わることになり、TOS変数をレジスタに割り当てることによってメモリアクセス回数が削減されるという効果も打ち消されてしまい、全体としての実行速度が向上されない。

#### （第 2 の従来技術の問題点）

上記第 2 の従来技術、つまり、ネイティブコード化と呼ばれている方法には、アーキテクチャの異なる実マシンを対象として仮想マシンプログラムを共有化することが困難になるという問題点がある。この従来技術では、仮想マシンプログラムの一部が特定の実マシンを対象とする実マシン命令で記述されるからである。そのために、例えば、ネットワーク上の異なるアーキテクチャの実マシンを備える 5 種類のコンピュータによって同じプログラムが共有される場合であれば、

それら5種類の実マシンを対象とする実マシンプログラムを準備しておく必要がある。

【0037】

また、コンピュータのシステム環境は個々に異なるために、たとえ同じアーキテクチャの実マシンを対象とする場合であっても、必ずしも仮想マシン命令よりも実マシン命令のほうが実行速度の点で優るとも言えない。例えば、プログラムのコードサイズが一般的に大きくなってしまいうRISC（縮小型命令セットコンピュータ）タイプの実マシンを対象とする場合では、そのメインメモリのサイズが充分でないときには仮想マシン命令を実マシン命令に置き換えることで仮想メモリのページ入れ替えが頻発し、却って実行速度が低下する結果となる。

（第3の従来技術の問題点）

上記第3の従来技術、つまり、その場式コンパイラと呼ばれる手法では、仮想マシンプログラムをコンパイルするのに要する時間が長くなってしまいうという問題点がある。

【0038】

その第1の理由は、対象とする実マシン固有のジャンプ先についての制限を満足するための処理が必要とされることである。例えば、ジャンプ先のアドレスがメインメモリのワード（基本語長）境界に制限される実マシンを対象とする場合には、仮想マシン命令をただ単に対応する実マシンプログラムに変換しただけでは、その制限を違反してしまう。

【0039】

図97は、この第1の理由を説明するためのサンプル仮想マシンプログラムのリストであり、図98は、そのフローチャートである。

このサンプル仮想マシンプログラムは、0から9までの合計10個の整数の和を算出するものであり、初期値の設定（ステップ7002、アドレス0～6）、計算の終了判断（ステップ7003、アドレス8～13）、加算と被加算値の設定（ステップ7004、アドレス15～29）及び終了処理（ステップ7005、アドレス31）からなる。

【0040】



図 99 は、この従来技術においてコンパイル時に使用される変換テーブルである。つまり、コンパイルの対象となる仮想マシン命令と、それと置き換えられる実マシンプログラム等との対応表である。なお、この変換テーブルには、参考のために、各実マシンプログラムのコードサイズも示されている。

図 100 は、図 99 に示された変換テーブルを用いて図 97 に示された仮想マシンプログラムをコンパイルした場合に得られる実マシンプログラムのコード配置の様子を図である。ここには、元の仮想マシンに対応する各実マシンプログラムの相対的な配置アドレスが示されている。

#### 【0041】

ここで、もし、対象とする実マシンが 2 ワード境界でしかジャンプ先を指定できないとするならば、図 100 から明らかなように、アドレス 13 の仮想マシン命令“Brz”で指定されるジャンプ先であるアドレス 31 の仮想マシン命令“Stop”及びアドレス 29 の仮想マシン命令“Br”で指定されるジャンプ先であるアドレス 15 の仮想マシン命令“Push”は、いずれも、実マシンプログラムにおける奇数アドレスに配置されるため、即ち、2 ワードの境界には配置されないため、ジャンプ先の制限に違反する。従って、そのような制限違反を犯さないための処理が必要とされる。

#### 【0042】

また、上記問題点を生じる第 2 の理由は、対象とする実マシンによっては分岐に伴う特別な処理が必要とされることである。例えば、米国 SPARC International, Inc. 社の SPARC (登録商標) アーキテクチャの CPU や米国 MIPS Technologies, Inc. 社の CPU など、一部のリスクアーキテクチャの CPU では、分岐命令の直後に配置されるいくつかの命令のみを特別なルールで実行するものがある。具体的には、分岐の成否にかかわらず後続する所定の命令を実行したり (遅延分岐; delayed branch)、分岐が成立した場合にのみ後続する所定の命令を実行したり (打ち消し分岐; cancelling branch) する。

#### 【0043】

このような実マシンを対象とする場合には、一定の解析を行った後に命令の配置をスケジューリングするか、NOP 等の無動作命令を分岐命令の直後に配置する

などの特別な処理が必要とされる。

(第4の従来技術の問題点)

上記第4の従来技術、つまり、予め仮想マシンプログラムを圧縮符号化しておく方法では、圧縮された仮想マシンプログラムにおける分岐命令の実行に伴って生じる不具合についての解決手段が示されていないという問題点がある。

【0044】

図101(a)は、この問題点を説明するための圧縮符号化テーブルの例であり、可変長符号9300aと仮想マシン命令9300bとを対応付けている。図101(b)は、図101(a)に示されたテーブルを用いて仮想マシン命令列A("babc")を符号化して得られる符号列である。

図101(b)に示された符号列は、その先頭から復号されるならば、元の仮想マシン命令列A("babc")に復元され得る。ところが、分岐命令などによって図101(b)の途中箇所Bに実行制御が移った場合において、その箇所Bから始まる符号列"0010110"に対して図101(a)に示されたテーブルを用いて復号した場合には、誤った仮想マシン命令列"aabc"に復号されてしまうという問題がある。

(第1～4の従来技術に共通の問題点)

上記第1～4の従来技術には、キャッシュ処理を効率化するための技術については提案されていないという共通の問題点がある。そのために、キャッシュメモリを備える実マシンやコンピュータの処理能力を生かした高速な仮想マシンの実現が期待されている。

【0045】

図102は、キャッシュメモリを備える仮想マシンにおいて生じうる問題点を説明するための図であり、仮想マシンのプログラムカウンタ6901と命令キャッシュ6902の構成を示すブロック図である。命令キャッシュ6902は、10番地分の命令列6903をキャッシュブロックとしてキャッシュメモリに存在する全てのキャッシュブロックを特定するためのアドレスを記憶するキャッシュテーブル6904を備える。

【0046】

図103は、図97に示された仮想マシンプログラムがキャッシュメモリに置かれた場合における各キャッシュブロックを区切る境界線A, B, Cを示す図である。本図に示された境界線Cから分かるように、単純に仮想マシンプログラムを一定長のキャッシュブロックに分割しただけでは、1個の仮想マシン命令”Br 8”のオペコード”Br”とオペランド”8”とが分断されてしまう。そのために、仮想マシンプログラムのキャッシュブロックへの分割において、プログラムカウンタ6901の値を変更する全ての仮想マシン命令について、キャッシュブロックを跨ぐような変更をしていないか判定しなければならず、処理が複雑化し、結果的にキャッシュを導入することで、仮想マシン全体の実行速度が低下してしまう。

【0047】

ここで、仮想マシンプログラム全体をキャッシュメモリに配置する方法や、その場式コンパイラによる仮想マシンプログラムの内容解析の結果を用いてキャッシュメモリに配置する方法が考えられる。しかし、前者の方法では、キャッシュメモリの利用効率が悪く、ネットワーク環境におけるデータ転送に要する時間が大きくなってしまいうという問題があり、また、後者の方法では、キャッシュメモリへの読み込み時間が長くなるという問題があり、いずれの方法によっても、仮想マシン全体としての実行効率は大幅に低下してしまう。

(発明の目的)

そこで、本発明はかかる問題点に鑑みてなされたものであり、従来よりも高速に仮想マシンプログラムを実行する仮想マシン、そのための仮想マシンコンパイラ（以下、仮想マシンと仮想マシンコンパイラとを併せて「仮想マシンシステム」と呼ぶ。）及びその場式コンパイラを提供することを目的とする。ここで、仮想マシンコンパイラとは、C言語等の高級言語で記述されたソースプログラムを仮想マシンプログラムに翻訳するプログラムである。

【0048】

この目的を達成するためのより具体的な目的は以下の通りである。

第1に、真のデータ依存関係の影響を吸収することで高速実行を可能とした仮想マシンシステムを提供する。

第2に、割込み処理に基づく実行効率の低下を回避することで高速実行を可能

とした仮想マシンシステムを提供する。

【0049】

第3に、複数の異なるアーキテクチャの実マシンを対象とする場合であっても、そのために全体としての実行速度を低下させることなく個々の実マシンの環境に応じたネイティブコード化が可能な仮想マシンシステム、即ち、実行速度を低下させることなく実マシンのアーキテクチャへの非依存性を高めた仮想マシンシステムを提供する。

【0050】

第4に、仮想マシンプログラムのキャッシュブロックへの分割に伴う実行効率の低下や、その場式コンパイラを採用した場合における複雑なアドレス解決に伴う実行効率の低下などを回避することで高速実行を可能としたキャッシュ機構に対応した仮想マシンシステムを提供する。

第5に、分岐命令が含まれている場合であっても圧縮された仮想マシンプログラムを正しく伸長できることが保証された高速実行可能な仮想マシンシステムを提供する。

【0051】

第6に、複雑なアドレス解決を行う必要のない高速動作が可能なその場式コンパイラを提供する。

【0052】

【課題を解決するための手段】

上記第1の具体的な目的を達成するために、本発明に係る仮想マシンは、実マシンによる制御の下で仮想マシン命令を実行するスタック型仮想マシンであって、データを後入れ先出し方式で一時的に記憶するためのスタック手段と、実行対象となる仮想マシン命令列と、それら仮想マシン命令それぞれに対応づけられた情報であって、対応づけられた仮想マシン命令に後続する仮想マシン命令が実行された場合の前記スタック手段でのデータの格納状態の変化を示す先行命令情報とを記憶する命令記憶手段と、前記命令記憶手段から次に実行すべき仮想マシン命令及び対応する先行命令情報を読み出す読み出し手段と、読み出された仮想マシン命令と先行命令情報との組み合わせに対応する演算処理を特定し実行する解

読実行手段とを備えることを特徴とする。

【0053】

また、本発明に係るコンパイラは、スタック型仮想マシンを対象とするコンパイラであって、ソースプログラムを前記仮想マシンが実行する仮想マシン命令列に変換する命令列変換手段と、変換によって得られた仮想マシン命令列を構成する仮想マシン命令それぞれについて、後続する仮想マシン命令が前記仮想マシンによって実行された場合の前記スタック手段でのデータの格納状態の変化を示す先行命令情報を生成する先行命令情報生成手段と、生成された先行命令情報と対応する仮想マシン命令とを関連付けて出力する関連付け手段とを備えることを特徴とする。

【0054】

上記第2の具体的な目的を達成するために、本発明に係る仮想マシンは、実マシンによる制御の下で仮想マシン命令を実行する仮想マシンであって、実行対象となる仮想マシン命令列を記憶する命令記憶手段と、前記命令記憶手段から次に実行すべき仮想マシン命令を読み出す読み出し手段と、読み出された仮想マシン命令に対応する演算処理を特定し実行する解読実行手段とを備え、前記解読実行手段は、読み出された仮想マシン命令が実行制御の流れを分岐させる分岐命令であるか否かを判定する分岐命令判定部と、分岐命令であると判定される度に、その分岐命令の実行に加えて、この仮想マシンへの割込み要求の発生の有無の検出と、発生している場合の割込み処理とを実行する割込み処理部とを有することを特徴とする。

【0055】

また、本発明に係る仮想マシンは、実マシンによる制御の下で仮想マシン命令を実行する仮想マシンであって、実行対象となる仮想マシン命令列を記憶する命令記憶手段と、前記命令記憶手段から次に実行すべき仮想マシン命令を読み出す読み出し手段と、読み出された仮想マシン命令に対応する演算処理を特定し実行する解読実行手段とを備え、前記解読実行手段は、読み出された仮想マシン命令が一定個数の仮想マシン命令の集まりをブロックとした場合の各ブロックを代表する仮想マシン命令に相当するか否かを判定するブロック判定部を備え、各ブロ

ックを代表する仮想マシン命令であると判定される度に、その仮想マシン命令の実行に加えて、この仮想マシンへの割込み要求の発生の有無の検出と、発生している場合の割込み処理とを実行する割込み処理部とを有するとすることもできる。

#### 【0056】

上記第3の具体的な目的を達成するために、本発明に係る仮想マシンは、実マシンによる制御の下で仮想マシン命令を実行する仮想マシンであって、実マシン命令からなる複数のサブプログラムを記憶する実マシンプログラム記憶手段と、実行対象となる仮想マシン命令列を記憶する第1領域と、前記実マシンプログラム記憶手段における各サブプログラムへのポインタを記憶する第2領域とを含む命令記憶手段と、前記命令記憶手段の第1領域から次に実行すべき仮想マシン命令を読み出す読み出し手段と、読み出された仮想マシン命令に対応する演算処理を特定し実行する解読実行手段とを備え、前記解読実行手段は、読み出された仮想マシン命令が前記第2領域に実行制御を移す命令であるか否かを判定する領域判定部と、前記第2領域に実行制御を移す命令であると判定された場合に、その移動先の第2領域の箇所に格納された前記ポインタが示す前記サブプログラムを実行するアドレス変換実行部とを有することを特徴とする。

#### 【0057】

ここで、前記命令記憶手段における第1領域と第2領域とは、一定のアドレス値を境界とする隣接した記憶位置に設けられ、前記領域判定部は、読み出された仮想マシン命令がサブプログラムを呼び出す命令である場合に、その呼び出し先アドレスと前記境界との大小関係を比較することによって前記判定を行うとすることもできる。

#### 【0058】

上記第4の具体的な目的を達成するために、本発明に係る仮想マシンは、実マシンによる制御の下で仮想マシン命令を実行する仮想マシンであって、実行対象となる仮想マシン命令列を記憶する命令記憶手段と、前記命令記憶手段から次に実行すべき仮想マシン命令を読み出す読み出し手段と、読み出された仮想マシン命令に対応する演算処理を特定し実行する解読実行手段とを備え、前記命令記憶

手段は、前記仮想マシン命令列を構成する基本ブロックそれぞれに対応する命令ブロックの集まりからなり、各命令ブロックは、その命令ブロックの前記命令記憶手段における先頭位置を特定するための識別子を格納した識別子領域と、対応する基本ブロックの非分岐命令だけを格納した非分岐命令領域と、対応する基本ブロックの分岐命令を格納した分岐命令領域とを含み、前記分岐命令領域に格納された分岐命令は、その分岐先が前記識別子によって指定され、前記解読実行手段は、読み出された仮想マシン命令が前記分岐命令である場合には、その分岐先として指定されている識別子に対応する命令ブロックの非分岐命令領域の先頭に実行制御を分岐させることを特徴とする。

## 【0059】

上記第5の具体的な目的を達成するために、本発明に係る仮想マシンは、実マシンによる制御の下で仮想マシン命令を実行する仮想マシンであって、実行対象となる仮想マシン命令列を圧縮符号で記憶する命令記憶手段と、前記命令記憶手段から次に実行すべき仮想マシン命令の圧縮符号を読み出し、対応する仮想マシン命令に復号する読み出し手段と、復号された仮想マシン命令に対応する演算処理を特定し実行する解読実行手段とを備え、前記命令記憶手段は、前記仮想マシン命令列を構成する基本ブロックそれぞれに対応する命令ブロックの集まりからなり、各命令ブロックは、その命令ブロックの前記命令記憶手段における先頭位置を特定するための識別子を格納した識別子領域と、対応する基本ブロックの非分岐命令だけを格納した非分岐命令領域と、対応する基本ブロックの分岐命令を格納した分岐命令領域とを含み、前記分岐命令領域に格納された分岐命令は、その分岐先が前記識別子によって指定され、前記解読実行手段は、復号された仮想マシン命令が前記分岐命令である場合には、その分岐先として指定されている識別子に対応する命令ブロックの非分岐命令領域の先頭に実行制御を分岐させることを特徴とする。

## 【0060】

上記第6の具体的な目的を達成するために、本発明に係るその場式コンパイラは、実マシンによる制御の下で仮想マシン命令を実行する仮想マシンと共に用いられ、実行対象となる一部の仮想マシン命令列をその実行に先立って実マシン命

令列に変換するその場式コンパイラであって、前記仮想マシン命令列を構成する仮想マシン命令それぞれについて、前記仮想マシン命令列を基本ブロックに分割した場合の各基本ブロックの先頭となる仮想マシン命令であるか否かを示すブロック先頭情報の入力を受け付けるブロック先頭情報獲得手段と、前記仮想マシン命令列を構成する仮想マシン命令それぞれについて、対応する実マシン命令列に変換する変換手段と、前記変換手段によって得られた実マシン命令列における実マシン命令であって、前記ブロック先頭情報獲得手段が受け付けたブロック先頭情報によって特定された基本ブロックの先頭となる仮想マシン命令に対応する実マシン命令が、前記実マシンによってアドレッシングされ得ない位置に配置される命令であるか否かを判定する分岐違反判定手段と、前記分岐違反判定手段によって前記実マシン命令が前記位置に配置されると判定された場合に、前記実マシン命令が前記位置に配置されないよう 1 個以上の無動作命令を前記実マシン命令列に追加挿入して出力する出力手段とを備えることを特徴とする。

【0061】

## 【発明の実施の形態】

以下、本発明の実施の形態について、図面を用いて説明する。

## (実施形態 1)

まず、真のデータ依存関係の影響を吸収する実施形態 1 に係る仮想マシンシステムについて説明する。

【0062】

図 1 は、本実施形態に係る仮想マシンシステムが動作するコンピュータのシステム（ハードウェア）構成図である。このコンピュータ 200 は、実マシン 201、メモリ 202、キーボード 203、マウス 204、ディスプレイ 206、ハードディスク 207、ネットワークカード 208 及びそれらを接続する内部バス 205a～205c から構成され、このハードウェア構成自体は、汎用のパーソナルコンピュータと異ならない。

【0063】

本実施形態に係る仮想マシン及びその仮想マシンコンパイラは、実マシン 201 用の命令で記述された実行形式のプログラムであり、ハードディスク 207 に



格納されているが、操作者や他のプログラムからの指示によってメモリ 202 にロードされ、実マシン 201 によって実行される。なお、実マシン 201 は、従来技術での説明と同様に、図 75 に示された実マシン命令を解釈・実行する CPU である。

#### （仮想マシンの構成）

図 2 は、本実施形態に係る仮想マシン 100 の構成を示す機能ブロック図であり、従来技術の説明における図 71 に対応する。この仮想マシン 100 は、先行命令情報格納部 101、命令格納部 102、デコード部 103、実行部 110 及びスタック 120 から構成される。

#### 【0064】

命令格納部 102 は、実行対象となる仮想マシンプログラムを保持する記憶領域であり、一方、先行命令情報格納部 101 は、その仮想マシンプログラムを構成する各仮想マシン命令に対応する先行命令情報を保持する記憶領域である。ここで、先行命令情報とは、仮想マシンプログラムにおいて対象とする仮想マシン命令の次に実行される仮想マシン命令がスタック 120 の階層を増加させる命令であるか減少させる命令であることを示す 1 ビットの情報（前者の場合の情報を「U」、後者の場合の情報を「D」と記す。）をいい、後述する本実施形態に係る仮想マシンコンパイラによってソースプログラムから仮想マシンプログラムと共に生成され得る。

#### 【0065】

図 3（a）及び（b）は、それぞれ先行命令情報格納部 101 及び命令格納部 102 に格納されている先行命令情報及び対応する仮想マシンコードの例を示す。ここでは、図 80（a）に示された仮想マシンプログラムと同一内容、即ち、「 $2 \times (3+4)$ 」を計算する仮想マシンプログラム及びその先行命令情報が格納されている。例えば、先行命令情報格納部 101 のアドレス 1 及び 2 には、命令格納部 102 の対応するアドレス 1 及び 2 に置かれた仮想マシン命令「Push 2」の次に実行される仮想マシン命令「Push 3」がスタック 120 の階層を増加させる旨の先行命令情報「U」が格納されている。

#### 【0066】

デコード部103は、先行命令情報格納部101に格納された先行命令情報を参照しながら命令格納部102に格納された仮想マシン命令を逐一に読み出して解読し、その結果を実行部110に出力するものであり、さらに、先行命令情報読込み部104、命令読込み部105、検索部106、プログラムカウンタ（PC）107及びデコードテーブル108から構成される。

#### 【0067】

PC107は、命令格納部102及び先行命令情報格納部101それぞれに格納された次に読み出すべき1対の仮想マシン命令及び先行命令情報のアドレス（これらのアドレスは、本実施形態では同一としている。）を保持するための記憶領域であり、実行部110によってそのアドレスが書き込まれることによって更新される。なお、PC107は、従来技術と同様に、物理的には、実マシン201のレジスタ2番（r2）に割り当てられている。

#### 【0068】

命令読込み部105は、PC107が示す命令格納部102のアドレスに置かれた仮想マシン命令を読み出して検索部106に送る。同様にして、先行命令情報読込み部104は、命令読込み部105に同期して動作し、PC107が示す先行命令情報格納部101のアドレスに格納された先行命令情報を読み出し、検索部106に送る。

#### 【0069】

デコードテーブル108は、この仮想マシン100が解読・実行する全ての仮想マシン命令（ここでは、図72に示された仮想マシン命令とする。）のオペコードと上述した2種類の先行命令情報との全ての組み合わせについて、マイクロプログラム記憶部111に格納された対応するマイクロプログラムへのジャンプアドレス及びその仮想マシン命令に伴うオペランドの数を予め記憶する。なお、全てのオペコードは1バイト長であり、オペランドの単位も1バイトである点は、従来技術と同じとする。

#### 【0070】

図4は、デコードテーブル108の内容を示す図であり、図73に示された従来技術におけるデコードテーブル4406に対応する。このデコードテーブル1

08には、従来のデコードテーブル4406と相違し、1種類の仮想マシン命令108aにつき、「U」と「D」の2種類の先行命令情報108bに対応するジャンプアドレス108c及びオペランド数108dが格納されている。例えば、1種類の仮想マシン命令"Push"について、先行命令情報108bが「U」の場合には、上向きPushを処理するマイクロプログラムへのジャンプアドレスが格納され、一方、先行命令情報108bが「D」の場合には、下向きPushを処理するマイクロプログラムへのジャンプアドレスが格納されている。

#### 【0071】

検索部106は、命令読み込み部105から送られてきた仮想マシン命令のオペコードと先行命令情報読み込み部104から送られてきた先行命令情報との組み合わせに対応するデコードテーブル108内のエントリを特定し、そのエントリに格納されたジャンプアドレスを読み出し、それをデコードデータとして実行部110に出力する。

#### 【0072】

実行部110は、検索部106から送られてきたデコードデータに従って、仮想マシン命令のオペレーションを実行するものであり、さらに、マイクロプログラム記憶部111及びスタックポインタ（SP）112を有する。

マイクロプログラム記憶部4411は、この仮想マシン100が解読・実行する全ての仮想マシン命令と上述した2種類の先行命令情報との全ての組み合わせに対応するマイクロプログラムを予め記憶する。これらマイクロプログラムの詳細については後述する。

#### 【0073】

SP112は、従来技術と同様に、スタック120の最上位のアドレスを保持する記憶領域であり、物理的には、実マシン201のレジスタ3番（r3）に割り当てられている。

スタック120は、実行部110による仮想マシンプログラムの実行のための一時的な後入れ先出し方式の記憶領域であり、さらに、TOS変数121、Second of Stack（SOS）変数122及びメモリスタック123から構成される。TOS変数121は、スタック120の最上位の値を格納する記憶域であり、物理的には、

実マシン201のレジスタ0番(r0)に割り当てられている。SOS変数122は、スタック120の最上位から2番目の値を格納する記憶域であり、物理的には、実マシン201のレジスタ4番(r4)に割り当てられている。メモリスタック123は、スタック120の最上位から3番目以降の値を格納する記憶域であり、物理的には、メモリ202に割り当てられている。

(マイクロプログラム記憶部111の内容)

図5(a)及び(b)は、それぞれ、マイクロプログラム記憶部111に格納されているマイクロプログラムのうち、上向き仮想マシン命令"Push"及び下向き仮想マシン命令"Push"に対応するもののリストを示す。同様に、図6(a)及び(b)は、それぞれ、上向き仮想マシン命令"Add"及び下向き仮想マシン命令"Add"に対応するもの、図7(a)及び(b)は、それぞれ、上向き仮想マシン命令"Mult"及び下向き仮想マシン命令"Mult"に対応するもののリストを示す。そして、図8(a)及び(b)それぞれは、図6(a)及び図7(a)に示された上向き仮想マシン命令のマイクロプログラムの後半に相当する部分、図6(b)及び図7(b)に示された下向き仮想マシン命令のマイクロプログラムの後半に相当する部分のリストである。なお、ここで用いられている実マシン命令の意味は、図75に示される通りである。

【0074】

これらマイクロプログラムのリストと図74(a)～(d)及び図82(a)～(d)に示された従来技術のものと比較して分かるように、本仮想マシン100のマイクロプログラム記憶部111に格納されたマイクロプログラムには以下の特徴がある。

つまり、第1の特徴は、同一の仮想マシン命令であっても先行命令情報が異なれば対応するマイクロプログラムの内容が異なることである。これは、直後に実行される仮想マシン命令によるスタック操作を予め考慮することによって、無駄なスタック操作や、真のデータ依存関係に基づくパイプラインの乱れを回避するためである。例えば、図5(b)に示されるように、仮想マシン命令"Push"のマイクロプログラムであるにも拘わらず、SOS変数122の値をメモリスタック123にプッシュする操作が記述されていないのは、この仮想マシン命令の先行命

令情報が下向き、即ち、次に実行される仮想マシン命令によってポップ操作が行われるので、無駄となってしまうことを避けるためである。

#### 【0075】

第2の特徴は、スタック120の最上位の記憶域(TOS変数121)だけでなく上位から2番目の記憶域(SOS変数122)についても、メモリ202ではなくレジスタに割り当てられていることである。これは、加算などの演算の対象となる2つの値をレジスタに保持することで、実マシン201とメモリ202間のデータ転送の回数を削減するためである。例えば、図6(a)に示されるように、加算だけのオペレーションであれば、レジスタとメモリ202間でのデータ転送は不要となる。

#### 【0076】

図9は、本仮想マシン100によって実行される仮想マシン命令が属する種別の変化を示す状態遷移図である。

ここで、各状態は、本仮想マシン100によって解釈・実行される全ての仮想マシン命令とその先行命令情報との組み合わせの全てについて、スタック120に対する操作内容の観点から分類した場合における各分類種別に相当する。各状態中に記された「X、Y(Z)」におけるX、Y、Zは、それぞれ、演算対象となるスタック格納値の数、演算後に増加するスタックの階層数、先行命令情報を示す。例えば、状態“2,-1(U)”は、2つの値を用いて演算し、その演算後にはスタックの階層数が1つ減少し、先行命令情報「U」が対応付けられた仮想マシン命令の全てを示す。具体的には、上向き仮想マシン命令“Add”などが該当する。なお、各状態の右下に記された式は、その状態に属する仮想マシン命令によるTOS変数121とSOS変数122の変化を示し、Xはオペランドを意味する。

#### 【0077】

また、図9において、遷移矢印線の種別は、遷移元の状態に属する仮想マシン命令に必要なオペレーションのうち、遷移先に依存することなく先行して行うことが可能な処理の内容を示す。従って、各状態から出ていく遷移矢印線の種別は全て同一である。例えば、状態“2,-1(D)”からの遷移先としては、“2,-1(U)”、“2,-1(D)”、“1,0(U)”、“1,0(D)”、“1,-1(U)”、“1,-1(D)”の6種類が考えられるが、いず

れに遷移する場合であっても、その遷移矢印線の種別が示す処理"Pop SOS"、即ち、メモリスタック 123 の最上位の値を SOS 変数 122 にポップすることが可能であることが分かる。なお、無条件分岐命令"Br"及び終了命令"Stop"は、TOS 変数 121 に対して空の演算を行う命令"1,0(U)"（又は"1,0(D)"）としている。

#### 【0078】

このように、図 9 に示された状態遷移図は、本仮想マシン 100 に係る全ての仮想マシン命令について先行して行うことが保証された処理内容を示す解析結果と言える。そして、この解析結果は、対応するマイクロプログラムに反映されている。つまり、マイクロプログラム記憶部 111 に格納された各マイクロプログラムには、この状態遷移図に示された対応する先行処理（対応する状態から出ていく遷移矢印線が示す処理）が組み込まれている。

#### （仮想マシンの動作）

以上のように構成された仮想マシン 100 の動作について説明する。

#### 【0079】

図 10 は、本仮想マシン 100 のデコード部 103 の動作を示すフローチャートであり、従来技術における図 76 に対応する。これらの図を比較して分かるように、このデコード部 103 の動作は、従来のデコード部 4402 と基本的な流れは同一であるが、先行命令情報読込み部 104 が命令読込み部 105 と同期して先行命令情報格納部 101 から先行命令情報を読み込む処理（ステップ 4906）が追加されている点と、テーブル検索（ステップ 4907）における詳細な動作内容において、従来と異なる。

#### 【0080】

図 11 及び図 12 は、それぞれ、図 10 に示されたテーブル検索（ステップ 4907）の詳細を示すフローチャートの前半部及び後半部であり、従来技術における図 77 に対応する。

これらの図を比較して分かるように、本実施形態におけるテーブル検索は、従来のものと基本的な流れは同一であるが、本仮想マシン 100 の検索部 106 が命令読込み部 105 から送られてくる仮想マシン命令のオペコードだけでなく先行命令情報読込み部 104 から送られてくる先行命令情報との組み合わせを考慮

し、それらが一致するデコードテーブル 108 のエントリに格納されたジャンプアドレス 108 c とオペランド数 108 d を参照してデコーデータを特定し実行部 110 に出力する処理（ステップ 5003、5007）が追加されている点で、従来と異なる。

#### 【0081】

図 13 は、図 3（a）及び（b）に示される先行命令情報及び仮想マシン命令が先行命令情報格納部 101 及び命令格納部 102 に格納されている場合に、それらに基づいてデコード部 103 が実行部 110 に順次出力するデコードデータを示し、従来技術における図 80（c）に対応する。先行命令情報と仮想マシン命令との組み合わせに対応するマイクロプログラムへのジャンプアドレスが出力されていることが分かる。

#### 【0082】

なお、実行部 110 の基本的な動作手順は、図 79 に示された従来技術に係るフローチャートに示される通りである。つまり、実行部 110 は、PC 107 と SP 112 とを初期化しておいた後に（ステップ 4702）、デコード部 103 から送られてくるデコードデータを読み込んで（ステップ 4704）、そのデコードデコードに含まれるジャンプアドレスが示すマイクロプログラム記憶部 111 内のマイクロプログラムに分岐し実行する動作（ステップ 4705）を繰り返す（ステップ 4703～4704）。

#### 【0083】

図 14（a）及び（b）は、それぞれ、図 13 に示されたデコードデータ列に従って実行部 110 が図 3（b）に示された仮想マシンプログラムを実行した場合における各仮想マシン命令が実行される前後での PC 107、SP 112、TOS 変数 121、SOS 変数 122 及びメモリスタック 123 の一連の変化の前半部と後半部を示す図であり、従来技術における図 81（a）及び（b）や図 83（a）及び（b）に対応する。ここで、本図の遷移矢印の中には、対応する先行命令情報（U または D）と仮想マシン命令との組がそれぞれスラッシュ「/」の左側及び右側に記されている。本図においても、PC が 9 に達した段階で、計算式  $2 * (3+4)$  の値が算出されていることが分かる。

【0084】

この図14（a）及び（b）において特徴的なことは、例えば、仮想マシン命令“U/Push 3”の実行後において既にSOS変数122の値がメモリスタック123の最上位に格納されている点、仮想マシン命令“D/Push 4”の実行後においてSP112及びメモリスタック123の内容が変化していない点などである。これらは、上述した解析に基づく先行処理が行われていることに起因する。

【0085】

図15～図18は、いずれも、本仮想マシン100が図3（b）に示された仮想マシンプログラムを実行した場合の実マシン201のパイプラインの流れ、より具体的には、アドレス7の下向き仮想マシン命令“Add”のマイクロプログラム（図6（b））の後半部分（図8（b）に示されたジャンプ処理）と、続くアドレス8の下向き仮想マシン命令“Mult”のマイクロプログラム（図7（b））の前半部分（乗算計算処理）におけるパイプラインの流れを示す。但し、図15～図18は、それぞれ、実マシン201が通常マシンであってメモリ参照を行ってから1クロック後にその参照値を使用することができる場合、実マシン201がスーパースカラマシンであって同様の場合、実マシン201が通常マシンであってメモリ参照を行ってから2クロック後にその参照値を使用することができる場合、実マシン201がスーパースカラマシンであって同様の場合におけるパイプラインの流れを示し、従来技術における図93～図96に対応する。

【0086】

図15と図93とを比較して分かるように、パイプラインの乱れは、実マシン命令“Jmp”の実行に起因する1個の命令のキャンセルだけとなり、処理全体は1クロックで完了する。つまり、実マシン201が通常マシンであって、かつ、メモリ参照を行ってから1クロック後にその参照値を使用することができる場合には、本仮想マシン100は、実行速度において、従来の仮想マシンと異ならない。

【0087】

また、図16と図94とを比較して分かるように、本仮想マシン100によれば、従来の仮想マシンで生じた第1及び第2の依存関係は吸収され、パイプライ



ンの乱れは、実マシン命令"Jump"の実行に起因する1個の命令のキャンセルだけとなり、処理全体は9クロックで完了する。つまり、実マシン201がスーパースカラマシンであって、かつ、メモリ参照を行ってから1クロック後にその参照値を使用することができる場合、本仮想マシン100によって、11クロック要する従来の仮想マシンに比べ、実行速度は22%だけ高速化される。

## 【0088】

これは、下向き仮想マシン命令"Add"のマイクロプログラムにおいて、次に実行すべき仮想マシン命令"Mult"の実行のための先行処理、即ち、メモリストック123からSOS変数122へのポップ (Load r4, [r3]) 及びSP112のデクリメント (Dec r3) を実行しているので、メモリ参照 (Load r1, [r2]) から分岐 (Jump r1) までに十分な時間が確保され、パイプラインの乱れが吸収されたからである。

## 【0089】

また、図17と図95とを比較して分かるように、本仮想マシン100によれば、同様の理由により、従来の仮想マシンで生じた第1及び第2の依存関係は吸収されるので、パイプラインの乱れは、実マシン命令"Jump"の実行に起因する1個の命令のキャンセルだけとなり、処理全体は11クロックで完了する。つまり、実マシン201が通常マシンであって、かつ、メモリ参照を行ってから2クロック後にその参照値を使用することができる場合、本仮想マシン100によって、13クロック要する従来の仮想マシンに比べ、実行速度は18%だけ高速化される。

## 【0090】

また、図18と図96とを比較して分かるように、本仮想マシン100によれば、同様の理由により、従来の仮想マシンで生じた第1の依存関係に基づくハザードの数が低減され、かつ、第2の依存関係は吸収されるので、パイプラインの乱れは、第1の依存関係に基づく1クロック分のハザードと実マシン命令"Jump"の実行に起因する1個の命令のキャンセルだけとなり、処理全体は10クロックで完了する。つまり、実マシン201がスーパースカラマシンであって、かつ、メモリ参照を行ってから2クロック後にその参照値を使用することができる場合

、本仮想マシン100によって、13クロック要する従来の仮想マシンに比べ、実行速度は30%だけ高速化される。

#### 【0091】

以上のように、本実施形態に係る仮想マシン100は、個々の仮想マシン命令の実行において、対応する先行命令情報を参照することで、真のデータ依存関係をもつ2つの実マシン命令を実行する間に、後続する仮想マシン命令の実行において必要とされるスタック処理を先行して実行する。これによって、真のデータ依存関係に基づくパイプラインの乱れが抑えられ、実行速度が向上する。

#### （仮想マシンコンパイラの構成）

次に、上記仮想マシン100を対象とする仮想マシンコンパイラについて説明する。

#### 【0092】

図19は、上記仮想マシン100を対象とする仮想マシンコンパイラ3400の構成を示す機能ブロック図である。本仮想マシンコンパイラ3400は、高級言語で記述されたソースプログラム3404を入力とし、上記仮想マシン100固有の仮想マシン命令（図72に示された仮想マシン命令）からなる命令列3405aと、それら仮想マシン命令に対応する先行命令情報の列3405bとを生成するクロスコンパイラであり、さらに、命令列変換部3402、先行命令情報生成部3401及び関連付け部3403から構成される。

#### 【0093】

命令列変換部3402は、ネットワークカード208やハードディスク207等から経路Sを経て入力されるソースプログラム3404を構文解析して言語変換することによって、上記仮想マシン100固有の仮想マシン命令からなる命令列を生成し、その命令列を経路C1及びC3を介して先行命令情報生成部3401及び関連付け部3403に順次出力する。

#### 【0094】

先行命令情報生成部3401は、命令列変換部3402から送られてきた個々の仮想マシン命令に対応する先行命令情報を特定し、経路C2を介して関連付け部3403に順次出力する。なお、命令列変換部3402及び先行命令情報生成部

3401は、対応する仮想マシン命令と先行命令情報との組が同期して関連付け部3403に到達するように、それらを出力する際のタイミングを調整している。

#### 【0095】

関連付け部3403は、命令列変換部3402から送られてくる仮想マシン命令と先行命令情報生成部3401から送られてくる先行命令情報とを組にして関連付けた後に、最終的な出力プログラム3405として、それぞれを経路D1及びD2を経てメモリ202やハードディスク207等に出力する。

図20及び図21は、経路Sを経て命令列変換部3402に入力されるソースプログラム3404のデータ構造を説明するための図であり、それぞれ、ソースプログラム3404の命令列（ここでは、“ $x := (1+2) * (3+4)$ ”）に対応する木構造及びその木構造を構成する各ノードのデータ構造を示す。各ノードは、ソースプログラム3404の命令列を構成する個々の命令に対応し、その種別を示す命令種別5201、左部分木へのポインタ5202及び右部分木へのポインタ5203から構成される。

#### （仮想マシンコンパイラの動作）

次に、以上のように表現されたソースプログラム3404が本仮想マシンコンパイラ3400に入力された場合の動作について説明する。

#### 【0096】

図22は、命令列変換部3402の動作手順の概略を示すフローチャートである。命令列変換部3402は、木構造で表現されたソースプログラム3404の命令列を読み込むと（ステップ5402）、まず、その木構造の枝を順次に辿るための計算用スタックを初期化する（ステップ5403）。そして、その木構造におけるルートノードへのポインタを変数ptrにセットした後に（ステップ5404）、その木構造全体に対応する仮想マシン命令列、即ち、仮想マシンコードを生成し（ステップ5405）、バイト単位で先行命令情報生成部3401及び関連付け部3403に送る。

#### 【0097】

図23は、図22におけるステップ5405の詳細を示すフローチャートであ

る。命令列変換部3402は、木構造における左部分木に位置するノードに対する変換処理（ステップ5603～5606）を右部分木に位置するノードに対する変換処理（ステップ5607～5610）に優先させて、各ノードの命令種別5201を数値又はアドレスとしてそのまま出力していくか又は対応する仮想マシンコードに変換するかの処理を繰り返す（ステップ5611～5613）。なお、全てのノードについて処理を繰り返すために、ステップ5605及びS5609において、この処理（ステップ5601～5614）自体を再帰的に呼び出している。

#### 【0098】

図24は、図23におけるステップ5613の詳細を示すフローチャートであり、図20に示されるサンプルプログラムに対応するものである。命令列変換部3402は、各ノードの命令種別5201の値kndによって、対応する4種類の仮想マシン命令"Push","Mult","Pop","Add"のいずれかのオペコードを生成していく（ステップ5901～5909）。

#### 【0099】

図25は、先行命令情報生成部3401の動作を示すフローチャートである。先行命令情報生成部3401は、命令列変換部3402から順次送られてくる仮想マシンコードをバイト単位で受け取り（ステップ5502）、その2バイト目以降の仮想マシンコードについて、それがオペランドであるか、仮想マシン命令"Push"のオペコードであるか、他の仮想マシン命令のオペコードであるかを判断し、それぞれに対応する先行命令情報Nextを関連付け部3403に出力していく（ステップ5503～5509）。なお、最後に出力する先行命令情報は、固定的に"U"としている（ステップ5510）。

#### 【0100】

図26は、関連付け部3403の動作を示すフローチャートである。関連付け部3403は、直前に処理した仮想マシンコードの先行命令情報を記憶する変数prv及び最終的に生成する先行命令情報及び仮想マシンコードのアドレスAddrを初期化した後に（ステップ6002）、命令列変換部3402から送られてくる仮想マシンコードが無くなるまで次の処理（ステップ6004～6010）を繰

り返す（ステップ6003）。

#### 【0101】

つまり、命令列変換部3402及び先行命令情報生成部3401からそれぞれ経路C1及びC2を経て、対応する1バイト分の仮想マシンコード及び先行命令情報Nextを受け取った後に（ステップ6004、6005）、その先行命令情報Nextが直前と同一である旨を示す“X”であるか否か判断することによって（ステップ6006）、その仮想マシンコードの先行命令情報nowを決定し（ステップ6007、6008）、決定した先行命令情報nowと上記仮想マシンコードとを対にしてアドレスAddrで示されるメモリ202等の出力領域における位置に出力する（ステップ6009、6010）。

#### 【0102】

このようにして、本仮想マシンコンパイラ3400によって、高級言語等で記述されたソースプログラム3404から、本実施形態に係る仮想マシン100をターゲットとする仮想マシンプログラム、即ち、図2に示された仮想マシン100の先行命令情報格納部101及び命令格納部102に格納すべき先行命令情報列及び仮想マシン命令列の組が生成される。

#### 【0103】

また、本実施形態に係る仮想マシンコンパイラ3400は、木構造で表現されたソースプログラム3404を入力としたが、本発明はこれに限定されるものではなく、C言語等のプログラミング言語で記述されたテキストそのものを入力としてもよい。そして、命令列変換部3402が、そのテキストを木構造や3オペランド方式等による中間コードに変換する前処理を行ってもよい。

#### （実施形態2）

次に、割込み処理に基づく実行効率の低下を回避した実施形態2に係る仮想マシンについて説明する。

#### （仮想マシンの構成）

図27は、本実施形態に係る仮想マシン3500の構成を示す機能ブロック図である。本仮想マシン3500は、命令格納部4401、デコード部3502、割込み制御部3510、実行部4410及びスタック4420から構成される。

【0104】

本図と図71とを比較して分かるように、本仮想マシン3500は、従来の仮想マシン4400とほぼ同一の構成要素を備える。同一の構成要素については、同一の符号を付し、その説明を省略する。

相違する点は、従来の仮想マシン4400が備える構成要素に加えて、さらに、この仮想マシン3500への割込み要求に応じた処理を実行制御するための割込み制御部3510を備えること、デコード部3502からの制御信号及びデコードデータは分岐命令検出部3505に出力されていること、実行部3515のマイクロプログラム記憶部4411には割込み処理のための実マシンプログラムである割込み処理プログラム3516が追加して格納されていることである。以下、従来の仮想マシン4400と相違する点を中心に説明する。

【0105】

割込み制御部3510は、本仮想マシン3500が分岐命令を解釈・実行するたびに割込み発生の有無を検出し、必要な割込み処理を実行部4410に行わせるよう制御するものであり、さらに、分岐命令検出部3505、割り込み命令挿入部3506及び割込み状態格納部3507から構成される。

分岐命令検出部3505は、検索部4405から信号線D1を介して送られてきたデコードデータが分岐命令"Br","Brz","Brnz","Call","Ret"のいずれかのマイクロプログラムへのジャンプアドレスであるか否か判断し、肯定的に判断した場合にのみ、信号線C2をオンにしてそのデコードデータを割り込み命令挿入部3506に出力し、否定的に判断した場合には、信号線C2をオフにしたままそのデコードデータを割り込み命令挿入部3506に出力する。

【0106】

割込み状態格納部3507は、本仮想マシン3500に対する割込み要求が発生しているか否か及び発生している場合の割込み種別を特定する状態変数IDを保持する記憶域であり、物理的には、メモリ202やネットワークカード208内のレジスタ等に割り当てられる。

割り込み命令挿入部3506は、分岐命令検出部3505が分岐命令を検出した旨の通知を信号線C2を介して受けると、その時点における割込み状態格納部3

507の状態変数IDを参照することで、割込みが発生しているか検査し、発生している場合に、その分岐命令のデコードデータの出力に先立ち、割込み処理を実行させるためのデコードデータ、即ち、マイクロプログラム記憶部4411内の割込み処理プログラム3516へのジャンプアドレスと上記状態変数IDとを実行部4410に出力する。

#### 【0107】

割込み処理プログラム3516は、割り込み命令挿入部3506から通知された状態変数IDに基づくメモリ202上のアドレスに格納された割込みベクタを読み出し、その割り込みベクタが示すジャンプ先のサブルーチンを処理するという割込み処理を行うための実マシンプログラムである。

#### （仮想マシンの動作）

以上のように構成された仮想マシン3500の動作について説明する。

#### 【0108】

図28は、デコード部3502の動作のうち、テーブル検索とデコードデータの出力処理の詳細を示すフローチャートであり、従来における図77に対応する。図77と異なる点は、デコードデータの出力先の変更に伴う処理（ステップ6108～6111）である。つまり、検索部4405は、読み込み部4403から送られてきた仮想マシン命令のオペコードに対応するジャンプアドレスをデコードテーブル4406から読み出し終わると（ステップ6106）、分岐命令検出部3505に対して、信号線C1をオンにしたまま信号線D1を介してそのジャンプアドレスをデコードデータとして出力する（ステップ6108～6110）。

#### 【0109】

図29は、分岐命令検出部3505の動作を示すフローチャートである。分岐命令検出部3505は、信号線D1を介して入力されたデコードデータを読み込んで一時記憶した後に（ステップ6202、6203）、そのデコードデータddataがマイクロプログラムへのジャンプアドレスであるか否かを信号線C1の状態によって判断する（ステップ6204）。

#### 【0110】

その結果、そのようなジャンプアドレスである場合にはさらに、そのジャンプ

アドレスが予め内部に記憶する分岐命令"Br","Brz","Brnz","Call","Ret"のマイクロプログラムのジャンプアドレスのいずれかと一致するか否かを判断し（ステップ6205）、一致するときのみ信号線C2をオンにして（ステップ6206）、一時記憶していたデコードデータddataを出力する（ステップ6206～6208）。

#### 【0111】

一方、他の場合においては（ステップ6204、6205）、信号線C2をオフにしたまま、デコードデータddataを出力する（ステップ6207、6208）。

図30は、割込み命令挿入部3506の動作を示すフローチャートである。割込み命令挿入部3506は、信号線D2を介して入力されたデコードデータを読み込んで一時記憶した後に（ステップ6302、6303）、そのデコードデータddata2が分岐命令のマイクロプログラムへのジャンプアドレスであるか否か信号線C2の状態によって判断する（ステップ6304）。

#### 【0112】

その結果、そのようなジャンプアドレスである場合には、割込み状態格納部3507に格納された状態変数IDを読み出し（ステップ6305）、その値から割込みが発生しているか否かを判断し（ステップ6306）、発生している場合にのみ、予め定められた割込み処理を実行させるためのデコードデータ（状態変数IDと割込み処理プログラム3516へのジャンプアドレス）を実行部4410に出力した後に（ステップ6307）、一時記憶していた分岐命令のデコードデータddata2を実行部4410に出力する（ステップ6308）。その結果、実行部3515は、分岐命令の実行に先立ち、状態変数IDに基づく割込み処理プログラム3516を実行する。

#### 【0113】

一方、信号線D2を介して入力されたデコードデータが分岐命令のマイクロプログラムへのジャンプアドレスでないと判断されたり（ステップ6304）、割込みが発生していないと判断された（ステップ6306）場合には、特別な処理を行なうことなく、単に、一時記憶していた分岐命令のデコードデータddata2を実



行部 4410 に出力する（ステップ 6308）。

【0114】

このようにして、本仮想マシン 3500 が 1 個の分岐命令を解読・実行するたびに、この仮想マシン 3500 への割込みが発生しているか否かが検査され、発生している場合には割込み処理が追加されて実行される。

なお、本実施形態に係る仮想マシン 3500 によれば、従来の仮想マシン 4400 に比べ、1 個の分岐命令につき、割込み処理のための 1 個の分岐命令が追加して実行されるので、メモリアクセスの回数が 1 回分余計に増える。しかしながら、通常のプログラムでは分岐命令から次の分岐命令までに平均 6 個の非分岐命令が存在するので、このような仮想マシンプログラムにおいては、本仮想マシン 3500 によって増加するメモリアクセスの回数は、1 命令あたりに換算すれば、わずか 0.2 以下である。従って、例えば、本実施形態の割込み処理機能を実施形態 1 に係る仮想マシン 100 に適用することで、TOS 変数によるメモリアクセス回数の削減の効果を打ち消してしまうことなく、全体として従来よりもメモリアクセス回数を削減させることができるので、実行速度の高速化に好適な割込み処理機能付き仮想マシンが実現される。

【0115】

以上のように、本実施形態に係る仮想マシン 3500 によれば、デコード部 3502 と実行部 4410 との間に割込み制御部 3510 が設けられ、分岐命令検出部 3505 によって分岐命令が解読実行される場合に限定して割込み発生の検出と割込み処理が行なわれるので、全ての仮想マシン命令ごとに行われる場合に比べて実行効率の低下が抑止され、適度な頻度で割込み検出処理が行なわれる。

【0116】

なお、本実施形態では、デコード部 3502 からのデコードデータを監視することで分岐命令の発生が検出されたが、これをデコード部 3502 に入力される仮想マシン命令（のオペコード）を監視することで同一の検出をしてもよい。

同様に、デコード部 3502 からのデコードデータを監視するのではなく、マイクロプログラム記憶部 4411 に記憶された分岐命令に対応するマイクロプログラムの中に、割り込み命令挿入部 3506 が有する手順を予め追加して記述し

ておく方法であっても同じ結果が得られる。

### （実施形態 3）

次に、実施形態 2 と別の方法によって割込み処理に基づく実行効率の低下を回避した実施形態 3 に係る仮想マシンについて説明する。

### （仮想マシンの構成）

図 31 は、本実施形態に係る仮想マシン 3600 の構成を示す機能ブロック図である。本仮想マシン 3600 は、命令格納部 4401、デコード部 3502、割込み制御部 3610、実行部 4410 及びスタック 4420 から構成される。

### 【0117】

本図と図 27 とを比較して分かるように、本仮想マシン 3500 は、実施形態 2 に係る仮想マシン 3500 とほぼ同一の構成要素を備える。相違する点は、実施形態 2 における分岐命令検出部 3505 がブロック化部 3605 に置き換えられていること、そのブロック化部 3605 についての接続関係である。以下、実施形態 2 に係る仮想マシン 3500 と相違する点を中心に説明する。

### 【0118】

ブロック化部 3605 は、本仮想マシン 3600 により解読された仮想マシンコードのブロック化、即ち、一定個数（例えば、10 バイト）の仮想マシンコードが解読されたか否かを繰り返し検出し、ブロック化（検出）する度にその旨を割り込み命令挿入部 3506 に通知する。

### （仮想マシンの動作）

以上のように構成された仮想マシン 3600 の動作について説明する。

### 【0119】

図 32 は、ブロック化部 3605 の動作を示すフローチャートである。ブロック化部 3605 は、信号線 D1 を介して入力されたデコードデータを読み込んで一時記憶した後に（ステップ 6402、6403）、その時点での PC 4404 の値を読み出す（ステップ 6404）。すなわち、デコード部 3502 から送られてきたデコードデータがいずれのアドレスに置かれた仮想マシンコードに対応するものかを参照する。

### 【0120】

そして、その値PCを予め記憶する定数bsizeで割った余りmを算出し（ステップ6405）、その余りmがゼロであるか否かを判断し（ステップ6406）、ゼロであるときにのみ信号線C2をオンにして（ステップ6407）、一時記憶していたデコードデータddataを出力する（ステップ6407～6409）。

一方、ゼロでない場合には（ステップ6406）、信号線C2をオフにしたまま、デコードデータddataを出力する（ステップ6408、6409）。

#### 【0121】

割り込み命令挿入部3506は、実施形態2の場合と同様にして、信号線C2がオンとなると、割り込み発生の有無を検査し、発生している場合にのみ、割り込み処理を実行させるためのデコードデータ、即ち、マイクロプログラム記憶部4411内の割り込み処理プログラム3516へのジャンプアドレスと上記状態変数IDとを実行部4410に出力する。

#### 【0122】

このようにして、本仮想マシン3600が一定個数bsizeの仮想マシンコードが解読される度に、この仮想マシン3600への割り込みが発生しているか否かが検査され、発生している場合には割り込み処理が追加されて実行される。つまり、本実施形態に係る仮想マシン3600によれば、割り込み検出処理が実行される頻度は、定数“bsize”で指定される数の仮想マシンコードにつき一回に限定される。

#### 【0123】

従って、実施形態2の場合と同様に、定数“bsize”を一定値以上とし、本実施形態の割り込み処理機能を実施形態1に係る仮想マシン100に適用することで、実施形態1に係る仮想マシン100におけるTOS変数によるメモリアクセス回数の削減の効果を打ち消してしまうことなく、全体として従来よりもメモリアクセス回数を削減させた高速な割り込み処理機能付き仮想マシンが実現される。

#### 【0124】

なお、本実施形態に係る仮想マシン3600によれば、ブロック化部3605はPC4404の値を参照しているが、PC4404自体は実マシン201のレジスタ2番（r2）に割り当てられているので、この参照によってはメモリアク

セスの回数が増加することはない。

また、本実施形態に係る仮想マシン 3600 によれば、定数 "bsize" を変更するだけで割込み検出処理の実行頻度を調整することが可能となり、メモリアクセスの回数を柔軟に制御できる仮想マシンが実現される。

#### 【0125】

なお、本実施形態のブロック化部 3605 は、デコード部 3502 から送られてきたデコードデータに対応する PC4404 の値と定数 bsize を比較したが、これに代えて、内部にカウンタを設けることによって、デコード部 3502 からの信号線 C1 がオンになった回数をカウントし、その値と定数 bsize とを比較してもよい。これによって、一定バイト数の仮想マシンコードではなく、一定個数の仮想マシン命令ごとに割込み検出処理が行なわれることになる。

#### 【0126】

さらに、本実施形態では、割込み制御部 3610 が独立してブロック化を行ったが、実行部 4410 が PC4404 を参照できることを考慮すれば、マイクロプログラム記憶部 4411 の内容として割込み制御部 3610 の処理手順を予め追加しておく方法であっても同じ結果が得られる。

#### (実施形態 4)

次に、実マシンのアーキテクチャへの非依存性を高めた実施形態 4 に係る仮想マシンについて説明する。

#### (仮想マシンの構成)

図 33 は、本実施形態に係る仮想マシン 3700 の構成を示す機能ブロック図である。本仮想マシン 3700 は、命令格納部 3701、デコード部 4402、実行部 3710 及びスタック 4420 から構成される。

#### 【0127】

本図と図 71 とを比較して分かるように、本仮想マシン 3700 は、従来の仮想マシン 4400 とほぼ同一の構成要素を備える。相違する点は、命令格納部 3701 に格納されている内容と、実行部 3710 に領域判定部 3704 とアドレス変換部 3705 が加えられていることと、実マシン関数記憶部 3706 が追加されていることである。以下、従来の仮想マシン 4400 と相違する点を中心に

説明する。

【0128】

実マシン関数記憶部3706は、実マシン命令で記述された関数（以下、「実マシン関数」と呼ぶ。）の集まり、つまり、仮想マシンプログラムで必要とされる定型的な処理を行う関数の集合を実行時ライブラリとして予め記憶しており、物理的には、メモリ202に割り当てられている。具体的には、後述する第0～ $(RM_{max}-RM_{min})$ 番までの合計 $(RM_{max}-RM_{min}+1)$ 個の実マシン関数を記憶している。

【0129】

命令格納部3701は、実行対象となる仮想マシンプログラムだけでなく、実マシン関数テーブルも予め保持している。ここで、実マシン関数テーブルとは、上記実マシン関数記憶部3706に格納された各実マシン関数それぞれへのポインタ（先頭アドレス）の集まりである。

図34は、命令格納部3701のメモリマップ、即ち、仮想マシン3700から見たメモリ領域の用途分割を示す。アドレス $VM_{min} \sim VM_{max}$ の領域は、仮想マシンプログラム6501、即ち、仮想マシン命令で記述された関数（仮想マシン関数）の集まりが配置される領域に割り当てられ、これに続くアドレス $RM_{min} \sim RM_{max}$ の領域は、実マシン関数テーブル6502が配置される領域に割り当てられている。なお、実マシン関数テーブル6502の領域は、仮想マシンプログラム6501の領域の直後に置かれている。つまり、アドレス $RM_{min}$ は、アドレス $VM_{max}+1$ に等しい。

【0130】

図35は、図34に示された実マシン関数テーブル6502の構造を示す。命令格納部3701のアドレス $RM_{min} \sim RM_{max}$ の領域には、各番地ごとに、第0～ $(RM_{max}-RM_{min})$ 番の実マシン関数へのポインタが格納されている。但し、各ポインタは、アドレスの逆順に対応づけて格納されている。例えば、第0番の実マシン関数は、アドレス $RM_{max}$ に置かれた仮想マシン関数が呼び出された場合に実行される関数であり、同様に、第 $(RM_{max}-RM_{min})$ 番の実マシン関数は、アドレス $RM_{min}$ に置かれた仮想マシン関数が呼び出された場合に実行さ

れる関数である。

#### 【0131】

領域判定部 3704 は、デコード部 4402 から出力されたデコードデータを監視することで、実行部 3710 による実行対象が関数呼び出し命令 "Call" である場合に、その命令の実行に先立ち、その呼び出し先が仮想マシンプログラム 6501 内であるか実マシン関数テーブル 6502 が置かれた領域内であるかを判定する。

#### 【0132】

アドレス変換部 3705 は、領域判定部 3704 によって実行対象の仮想マシン命令が実マシン関数テーブル 6502 の領域内への関数呼び出し命令 "Call" であると判定された場合にのみ、その仮想マシン命令 "Call" の実行に代えて、そのジャンプアドレスに対応する実マシン関数テーブル 6502 内の関数ポインタが示す実マシン関数記憶部 3706 内の実マシン関数を直接実マシン 201 に実行させる。

(仮想マシンの動作)

以上のように構成された仮想マシン 3700 の動作について説明する。

#### 【0133】

図 36 は、本仮想マシン 3700 の実行部 3710 の動作、特に、デコード部 4402 から関数呼び出し命令 "Call" のデコードデータが送られてきた場合の領域判定部 3704 とアドレス変換部 3705 の動作を中心としたフローチャートである。

領域判定部 3704 は、信号線 R の状態及び検索部 4405 からのデコードデータを監視することによって、デコード部 4402 から関数呼び出し命令 "Call" のオペランドが送られてきたことを知ると、その仮想マシン命令の実行に先立ち、そのオペランドが示す呼び出し先 Jaddr がアドレス RMmin ~ RMmax の範囲、即ち、実マシン関数テーブル 6502 が置かれた領域内であるかを判定する（ステップ 6802 ~ 6804）。

#### 【0134】

その結果、その呼び出し先 Jaddr が領域内であると判定された場合には、アド

レス変換部3705は、上記逆順に基づいて、呼び出し先Jaddrに対応する実マシン関数テーブル6502へのインデックスidxを算出し（ステップ6805）、そのインデックスidxが示す実マシン関数テーブル6502のエントリに格納されたポインタptrを読み出す（ステップ6806）。そして、実行部3710は、元の仮想マシン命令"Call"に代えて、ポインタptrが示す実マシン関数記憶部3706内の実マシン関数を直接実行する（ステップ6807）。

#### 【0135】

一方、領域判定部3704によって関数呼び出し命令"Call"の呼び出し先Jaddrが実マシン関数テーブル6502の領域外であると判定された場合には、実行部3710は、通常の関数呼び出し処理を続行する（ステップ6808～6810）。つまり、実行部3710は、戻り番地を記憶した後に（ステップ6808、6809）、その呼び出し先Jaddrに置かれた仮想マシン関数を実行する（ステップ6810）。

#### 【0136】

このようにして、仮想マシン命令"Call"の呼び出し先Jaddrが仮想マシンプログラム6501の領域に属する場合には、そのまま仮想マシン関数が呼び出され、一方、実マシン関数テーブル6502の領域に属する場合には、対応する実マシン関数が実行される。

ここで、図34に示されたメモリマップから分かるように、関数呼び出し命令"Call"に対して仮想マシン関数を実行させるか実マシン関数を実行させるかは、2つの領域6501、6502を区切る境界線を移動させることによって容易に変更することができる。例えば、境界線となるアドレスVMmaxを下げた場合には、これに伴ってアドレスRMminも下がるので、たとえ同一アドレスJaddrへの関数呼び出し命令"Call"であっても、それまでの仮想マシン関数の実行に代えて実マシン関数を実行させることが可能となる。同様に、境界線のアドレスVMmaxを上げた場合には、それまでの実マシン関数の実行に代えて仮想マシン関数を実行させることが可能となる。

#### 【0137】

以上のように、本実施形態に係る仮想マシン3700によれば、わずか1つの

パラメータ VMmax の値を変更することによって、仮想マシン関数への呼び出しをそのまま実行させたり実マシン関数に置き換えて実行させたりする制御が可能となる。従って、本仮想マシン 3700 は、複数の異なる種別の実マシンやコンピュータ環境の下で動作させる仮想マシンとして好適なアーキテクチャを備えていると言える。複数の異なるアーキテクチャの実マシンやコンピュータそれぞれに応じて仮想マシンプログラムの一部を実マシン関数に置き換えておく場合には、それらアーキテクチャごとに仮想マシン関数として実行させる箇所と実マシン関数として実行させる箇所とを容易に切り分けられるからである。

#### 【0138】

これによって、実行速度を低下させることなく、実マシンのアーキテクチャへの非依存性を高めた仮想マシンが実現される。

なお、本実施形態では、命令格納部 3701 のアドレス VMmin~VMmax の領域には、仮想マシンプログラム 6501 だけが配置されていたが、本発明はこれに限られず、例えば、図 37 に示されるように、各アドレスごとのメモリ属性 6701 (V 又は R) と、その属性に対応するデータ (仮想マシンプログラム) 又は実マシン関数テーブルへのインデックスを格納してもよい。これによって、境界線 VMmax を移動させることなく、同一アドレスへの仮想マシン関数の呼び出しに対してそのまま仮想マシン関数として実行させるか実マシン関数を直接実行させるかの切り替えが可能となる。

#### (実施形態 5)

次に、仮想マシンプログラムのキャッシュブロック化やその場式コンパイラのコンパイル時間が短縮化される実施形態 5 に係る仮想マシンシステムについて説明する。

#### (仮想マシンの構成)

図 38 は、本実施形態に係る仮想マシン 3800 の構成を示す機能ブロック図である。本仮想マシン 3800 は、命令格納部 3801、デコード部 3802、実行部 3810 及びスタック 4420 から構成される。

#### 【0139】

本図と図 71 とを比較して分かるように、本仮想マシン 3800 は、従来の仮



想マシン4400とほぼ同一の構成要素を備える。相違する点は、命令格納部3801に格納されている内容と、PC3804の構成と、実行部3810に分岐先変換部3811が追加されていることである。以下、従来の仮想マシン4400と相違する点を中心に説明する。

#### 【0140】

命令格納部3801は、実行対象となる仮想マシンプログラムを命令ブロックと呼ばれる単位に区分して記憶しているものであり、各命令ブロックを記憶する複数の命令ブロック格納領域3852a～3852dからなる。

ここで、命令ブロックとは、本仮想マシンプログラムを基本ブロックに分割し、得られた基本ブロックそれぞれにユニークな識別子を付与し、それら基本ブロックを論理的に接続するための分岐命令を後続位置に追加したものをいい、後述する本仮想マシン3800固有のコンパイラによって生成される。なお、基本ブロックとは、先頭に配置された命令をそのブロックへの唯一の入口とし、最後部に配置された命令をそのブロックからの唯一の出口とする命令列をいう。また、本実施形態では、識別子は、命令ブロック格納領域における各命令ブロックの先頭位置を特定するアドレス情報からなる。

#### 【0141】

命令ブロック格納領域3852a～3852dそれぞれは、各命令ブロックの識別子を格納している識別子格納領域3853aと、各命令ブロックに属する仮想マシン命令のうち分岐命令を除くもの（以下、「非分岐命令」と呼ぶ。）だけを格納している非分岐命令格納領域3854aと、その分岐命令だけを格納している分岐命令格納領域3855aとからなる。

#### 【0142】

図39は、命令格納部3801に格納されている仮想マシンプログラムの格納状態の例を示す図であり、図97に示されたサンプル仮想マシンプログラムが格納されている場合に相当する。

本図に示されるように、上記仮想マシンプログラムは5個の命令ブロック3852a～3852dに分割され、各命令ブロック3852a～3852dは、その命令ブロックの識別子3853a～3853dと、各命令ブロックに属する基

本ブロックの非分岐命令だけからなる部分 3854 a～3854 d と、その基本ブロックの最後部に配置された分岐命令及び次の命令ブロックに接続するための分岐命令とからなる部分 3855 a～3855 d とから構成される。

#### 【0143】

なお、図 39 及び図 97 それぞれに示された仮想マシンプログラムは、図 72 に示された各仮想マシン命令の意味から明らかなように、それらプログラムの制御フローは図 40 に示される通りであり、実質的に同一の処理内容を行うことは明らかである。

PC 3804 は、さらに、識別子セグメントレジスタ 3804 a とオフセットカウンタ 3804 b とから構成される。識別子セグメントレジスタ 3804 a は、次に読み出すべき命令格納部 3801 中の仮想マシンコードが属する命令ブロックの識別子に相当するセグメントアドレス（以下、「識別子セグメント」と呼ぶ。）を保持し、オフセットカウンタ 3804 b は、その仮想マシンコードの命令ブロックにおけるオフセットを保持している。

#### 【0144】

なお、本仮想マシン 3800 は、図 41 に示されるように、16 ビットによるアドレッシングを行い、その上位 8 ビットを識別子セグメント、下位 8 ビットをオフセットとしている。つまり、識別子セグメントレジスタ 3804 a には 8 ビットの識別子セグメントが格納され、オフセットカウンタ 3804 b には 8 ビットのオフセットが格納され、これらが連結された 16 ビットのアドレスによって、命令格納部 3801 中の 1 個の仮想マシンコードが特定される。

#### 【0145】

分岐先変換部 3811 は、実行部 3810 において分岐命令を実行する場合に、その分岐命令のオペランド、即ち、分岐先となる命令ブロックの識別子を、識別子セグメントとオフセットとの組み合わせに変換し、それぞれの値を PC 3804 に格納することで更新する。

#### （仮想マシンの動作）

以上のように構成された仮想マシン 3800 の動作について説明する。

#### 【0146】

デコード部 3802 及び実行部 3810 は、従来の仮想マシン 4400 とほぼ同様の動作をする。異なる点は、通常の順次実行においては、実行部 3810 によって PC 3804 のオフセットカウンタ 3804 b だけがインクリメントされる点と、分岐命令の実行においては、分岐先変換部 3811 によって PC 3804 の識別子セグメントレジスタ 3804 a とオフセットカウンタ 3804 b が更新される点である。

#### 【0147】

図 4 2 は、実行部 3810 における分岐先変換部 3811 の動作を示すフローチャートである。分岐先変換部 3811 は、デコード部 3802 から送られたきた分岐命令のオペランド、即ち、8 ビットで表現された命令ブロックの識別子 Jaddr を獲得すると（ステップ 8102）、それを分岐先の識別子セグメントとすると共にオフセットをゼロとして 16 ビットの物理アドレスを生成し、それぞれで PC 3804 の識別子セグメントレジスタ 3804 a 及びオフセットカウンタ 3804 b を更新する（ステップ 8103）。

#### 【0148】

図 4 3 は、図 3 9 に示された仮想マシンプログラムにおける論理的なアドレスと識別子とを物理的なアドレスに置き換えたものであり、分岐先変換部 3811 によるアドレス変換の結果を示す。例えば、図 3 9 における識別子番号 1 の命令ブロックにおける分岐命令 "Brz" のオペランド "x03" は、分岐先変換部 3811 によって識別子番号 3 の命令ブロックの先頭への物理的な番地 "x0300" に変換される。

#### 【0149】

以上のようにして、実行部 3810 は、分岐命令を実行するごとに、そのオペランドで指定された命令ブロックの先頭に分岐する制御を行う。このようにして、本仮想マシン 3800 は、命令ブロックに分割されて格納されている仮想マシンプログラムに対して、命令ブロックに分割されなかった場合と実質的に同じ処理手順により、解説・実行を行う。

#### （仮想マシンコンパイラの構成）

次に、本仮想マシン 3800 を対象とする仮想マシンコンパイラについて説明

する。

#### 【0150】

図44は、実施形態5に係る仮想マシンコンパイラ7660の構成示すブロック図である。本仮想マシンコンパイラ7660は、C言語等の高級言語で記述されたソースプログラム7650を入力とし、これを上記仮想マシン3800の命令格納部3801に格納するのに適した形式、つまり、命令ブロック集合7651に変換するコンパイラであり、中間命令列変換部7661、生成部7662及びブロック化部7663から構成される。

#### 【0151】

中間命令列変換部7661は、入力されたソースプログラムを構文解析した後に最適化のために一時的な中間コードに展開し、生成部7662は、中間命令列変換部7661により展開された中間コードを、例えば図97に示されるような仮想マシンプログラム7664のコードに変換する。これら中間命令列変換部7661及び生成部7662は、従来の一般的な仮想（又は実マシン）コンパイラが有する機能と異ならない。

#### 【0152】

ブロック化部7663は、生成部7662から生成された仮想マシンプログラム7664を上記命令格納部3801に格納可能な命令ブロックの集まりに変換するものであり、その際の主な処理は、基本ブロックへの分割と、その分割に伴うアドレス解決である。ここで、アドレス解決とは、仮想マシンプログラム7664中の分岐命令で用いられていた分岐先を命令ブロックの識別子IDに置き換える処理である。

#### 【0153】

そして、このブロック化部7663は、そのアドレス解決のために、一時的な変数テーブルとして、分岐先変更テーブル7663aを作成し用いる。

図45は、分岐先変更テーブル7663aの構造を示す図である。

本テーブルの各行（エントリ）は、生成される個々の命令ブロック及びブロック化部7663に入力された仮想マシンプログラム7664中の個々の分岐命令それぞれに対応して作成されるものである。各エントリにおいて、「コード位置

」は、当該命令ブロックの先頭命令又は当該分岐命令の仮想マシンプログラム7664上でのアドレス、「登録フラグ」は、当該分岐命令についてのアドレス解決が終了したか否かのフラグ、「参照位置識別子」及び「参照位置offset」は、当該命令ブロックに分岐する分岐命令又は当該分岐命令が置かれている命令ブロックの識別子及びオフセットを示す。

(仮想マシンコンパイラの動作)

図46は、本仮想マシンコンパイラ7660の特徴的な処理、即ち、ブロック化部7663の動作を示すフローチャートである。まず、ブロック化部7663は、命令ブロック集合7651として生成する命令ブロックの識別子IDと、各命令ブロックにおける相対的な命令格納位置を示すポインタoffsetと、仮想マシンプログラム7664から順次に読み出す1バイトの仮想マシンコードVCの位置を示すカウンタPCと、分岐先アドレスを変更するための分岐箇所の個数を示すカウンタRcountとを初期化する(ステップ7602~7603)。

【0154】

ブロック化部7663は、基本的には、カウンタPCをインクリメントしながら仮想マシンプログラム7664から1バイトずつ仮想マシンコードVCを読み出し、その仮想マシンコードVCが属すべき命令ブロックの識別子ID及びその命令ブロック中での相対位置であるポインタoffsetと共に、命令ブロック集合7651として出力する(ステップ7604~7611)。

【0155】

このとき、仮想マシンコードVCが基本ブロックの先頭であるか否かの判断(ステップ7607)と、分岐命令であるか否かの判断を行い(ステップ7608)、それぞれ肯定的に判断された場合には、対応する特別な処理(ステップ7701~7704及びステップ7609)を行う。

図47は、図46におけるステップ7607、即ち、仮想マシンコードVCが基本ブロックの先頭とすべきものか否かの判断処理の詳細を示す。つまり、仮想マシンコードVCが分岐先の命令及び分岐命令の直後に配置された命令のいずれかに該当する場合には、その仮想マシンコードVCは基本ブロックの先頭と判断する(ステップ7302~7306)。

## 【0156】

図46に示されるように、仮想マシンコードVCが基本ブロックの先頭であると判断すると、新たな命令ブロックを生成するために識別子IDを更新すると共に（ステップ7701）、直前の命令ブロック（識別子ID）の最後部と次の命令ブロック（識別子NID）の先頭とを接続するための無条件分岐命令を生成する（ステップ7702）。そして、この新たな命令ブロックに属する仮想マシンコードを生成するための準備をした後に（ステップ7703）、この新たな識別子NIDの付与に伴うアドレス解決を行う（ステップ7704）。

## 【0157】

また、ステップ7608において、仮想マシンコードVCが分岐命令であると判断した場合には、その分岐先を適切に変更するためのアドレス解決を行う（ステップ7609）。このブロック化部7663による分割処理と新たな分岐命令の追加処理などによって、元の仮想マシンプログラム7664における各仮想マシン命令の配置位置が変動するからである。

## 【0158】

図48は、図46におけるステップ7704、即ち、新たな命令ブロックの識別子NIDの付与に伴うアドレス解決の詳細を示す。ここでは、ブロック化部7663は、この新たな命令ブロックの識別子NIDの付与に伴って初めてアドレス解決が可能な分岐命令が発見された場合には、そのアドレス解決を行い（ステップ7905～7910）、そうでない場合には、以降の処理においてこの命令ブロックに分岐する分岐命令のアドレス解決を行うために、分岐先変更テーブル7663aへの追加登録をする（ステップ7913、7914）。

## 【0159】

図49は、図46におけるステップ7609、即ち、仮想マシンプログラム7664中の分岐命令で指定されていた分岐先のアドレス解決の詳細を示す。ここでは、ブロック化部7663は、この分岐命令が前方への分岐、即ち、分岐先変更テーブル7663aに既に登録された命令ブロックへの分岐である場合には、その分岐先を命令ブロックの識別子に置き換えることによってアドレス解決を行い（ステップ7802～7809、7812）、そうでない場合には、アドレス

未解決として新たなエントリを分岐先変更テーブル7663aに追加登録する（ステップ7810、7811）。

【0160】

以上のように、仮想マシンコンパイラ7660によって、高級言語で書かれたソースプログラムは、一旦、図97に示されるような一般的な仮想マシンプログラム7664に変換された後に、基本ブロックに分割されて識別子が付与され、さらに、各基本ブロックを接続するための分岐命令が追加された後に、識別子付与に伴うアドレス解決が行なわれることで、本実施形態に係る仮想マシン3800の実行対象となる命令ブロック集合7651に変換される。

（考察）

このように、本実施形態に係る仮想マシン3800及び仮想マシンコンパイラ7660によって、実行対象となる仮想マシンプログラムは、図97に示されるような従来形式のまま命令格納部3801に配置され実行されるのではなく、基本ブロックに分割された状態で命令格納部3801に配置され実行されることになるが、そのことの技術的意義について考察する。

【0161】

まず、その場式コンパイラを採用した場合のコンパイル時間という観点から考察する。

上述したように、従来のその場式コンパイラによれば、仮想マシンプログラムにおける各分岐先について、その位置が一定の制限を違反していないかどうかの解析が必要とされ、違反している場合には、その分岐先を移動させる等の処理が必要とされた。ところが、本仮想マシンシステムによれば、いずれの分岐先も、必ず、各命令ブロックの先頭であることが保証されている。従って、従来において必要とされた、分岐先についての上記処理のほとんどは不要となる。

【0162】

また、従来のその場式コンパイラによれば、遅延分岐等のために、分岐命令の後続位置に配置する命令、即ち、分岐の正否の影響を受けない命令を特定する等の処理が必要とされた。ところが、本仮想マシンシステムによれば、命令格納部3801に格納された仮想マシンプログラムは、各命令ブロック中において非分

岐命令格納領域と分岐命令格納領域とに分けられており、さらに、分岐命令格納領域においては1個の分岐命令のあとには高々1個の分岐命令だけが後続することが保証されている。従って、従来において必要とされた、遅延分岐等についての上記処理のほとんどは不要となる。

#### 【0163】

次に、仮想マシンのキャッシュ機構への対応化という観点から考察する。

従来のようなキャッシュ機構への対応化によれば、仮想マシンプログラムをキャッシュブロックに分割する際に、プログラムカウンタを変更する全ての仮想マシン命令についてキャッシュブロックの境界をまたいで変更していないか判断する必要があった。ところが、本仮想マシンシステムによれば、命令格納部3801に格納された仮想マシンプログラムを命令ブロック単位でキャッシュさせるならば、キャッシュの境界をまたいでプログラムカウンタを変更する仮想マシン命令は、分岐命令格納領域3855aに属する分岐命令だけとなる。

#### 【0164】

図50は、本実施形態に係る仮想マシン3800において、命令ブロック単位でキャッシュさせた場合のPC3804、命令ブロック格納領域3852a～3852d及びキャッシュテーブル8404の関係を示す図であり、従来における図102に対応する。従来では、10番地分の命令列6903がキャッシュブロックとして命令キャッシュ6902に置かれたが、本仮想マシン3800の場合には、命令ブロック3852a～3852dの単位で命令キャッシュ8402に配置され、それらの識別子がキャッシュテーブル8404で管理される様子が示されている。

#### 【0165】

図51は、本実施形態に係る仮想マシン3800において、命令ブロック単位でキャッシュさせた場合の実行部3810による分岐命令の実行処理を示すフローチャートであり、キャッシュ機構に対応させていない場合の図42に対応する。これらを比較して分かるように、本仮想マシン3800をキャッシュ機構に対応した仮想マシンとするには、実行部3810が、キャッシュテーブル8404内の識別子を参照することで命令ブロック単位でキャッシュヒットの有無を判断



し（ステップ 8504）、ミスヒット時には命令キャッシュ 8402 への読込みを行えばよい（ステップ 8505）。

#### 【0166】

このように、命令ブロック単位で仮想マシンプログラムをキャッシュすることにより、従来において必要とされたキャッシュブロックの境界に伴う判断処理などが不要になる。そして、ミスヒットに伴うキャッシュへの読込みが発生した場合であっても、もともとの仮想マシンプログラムが予め命令ブロック単位で分割格納されているので、読込み時の負荷も削減される。

#### 【0167】

以上のように、本実施形態に係る仮想マシンシステムでは、ソースプログラムは一般的な仮想マシンプログラムに変換された後にさらに基本ブロックを単位とする命令ブロックに分割されて命令格納部 3801 に格納され、全ての分岐命令の分岐先は命令ブロックの識別子にて指定されている。これにより、その場式コンパイラでの分岐先命令のアドレス解析処理は単純化され、コンパイル時間が短縮される。また命令キャッシングを命令ブロック単位で行うことで、キャッシュ境界に伴う判定処理は簡略化され、仮想マシンにキャッシュを導入した場合の実行効率の低下は、従来に比較して抑えられる。

#### 【0168】

なお、本実施形態の仮想マシンコンパイラ 7660 は、中間命令列変換部 7661 及び生成部 7662 を備えたが、これらに代えて、ソースプログラムから仮想マシンプログラムを生成する一般的なコンパイラを採用してもよいことはいまでもない。

#### （実施形態 6）

次に、上記実施形態 5 に係る仮想マシンにおける解読処理の高速化を図った実施形態 6 に係る仮想マシンについて説明する。

#### （仮想マシンの構成）

図 52 は、本実施形態に係る仮想マシン 3900 の構成を示す機能ブロック図である。本仮想マシン 3900 は、命令格納部 3901、デコード部 3902、実行部 3810 及びスタック 4420 から構成される。

## 【0169】

本図と図38とを比較して分かるように、本仮想マシン3900は、実施形態5に係る仮想マシン3800とほぼ同一の構成要素を備える。相違する点は、命令格納部3901に格納されている内容と、デコード部3902にカレントフラグ記憶部3907が追加されていること、命令読み込み部3903の機能と、実行部3910にカレントフラグ読出制御部3912が追加されていることである。以下、実施形態5に係る仮想マシン3800と相違する点を中心に説明する。

## 【0170】

命令格納部3901は、実施形態5の命令格納部3801と比較し、予め、実行対象となる仮想マシンプログラムを複数の命令ブロック3952a～3952dの単位で記憶している点において共通するが、各命令ブロック格納領域3952a～3952dには、当該命令ブロックの非分岐命令格納領域及び分岐命令格納領域（これらを併せて「仮想マシンコード領域」と呼ぶ。）に格納されている全ての仮想マシンコードに対応するデコードデータ列（以下、「デコード命令列」と呼ぶ。）を格納するためのデコード命令列格納領域3956a～3956dが設けられている点において異なる。

## 【0171】

図53（a）～（c）は、命令格納部3901に格納されている仮想マシンプログラムの格納状態の例を示す図であり、図97に示されたサンプル仮想マシンプログラムが格納されている場合に相当する。

本図に示されるように、各命令ブロック格納領域3952a～3952cに設けられたデコード命令列格納領域3956a～3956cは、さらに、上記デコード命令列を格納するための実マシンコード領域8607a～8607cと、当該実マシンコード領域8607a～8607cにデコード命令列が格納されているか否かを示すフラグを格納するフラグ領域8605a～8605cとから構成される。例えば、図53（b）に示された命令ブロック格納領域3952bは、実マシンコード領域8607bにデコード命令列が存在しないので、その旨（空）のフラグがフラグ領域8605bに格納され、一方、図53（c）に示された命令ブロック格納領域3952cは、実マシンコード領域8607cにデコード

命令列が存在するので、その旨（あり）のフラグがフラグ領域 8605c に格納されている。

#### 【0172】

なお、各実マシンコード領域に格納すべきデコード命令列は、例えば、実施形態 5 に係る仮想マシン 3800 を用いることによって予め得ることができる。このデコード命令列は、各命令ブロックの仮想マシンプログラムを実施形態 5 に係る仮想マシン 3800 が実行した場合に、そのデコード部 3802 が実行部 3810 に出力するデコードデータ列に等しいからである。

#### 【0173】

また、各命令ブロックにおいて、仮想マシンコード領域 3954a～3954d、3955a～3955d に置かれている個々の仮想マシン命令と、実マシンコード領域 8607a～8607d に置かれている対応するデコードデータとは、一定のオフセットだけ離れたアドレスの位置に配置されている。

カレントフラグ記憶部 3907 は、本仮想マシン 3900 の実行対象となっている命令格納部 3901 中の命令ブロックのフラグ領域に格納されているフラグをカレントフラグとして保持するための一時的な記憶領域である。

#### 【0174】

命令読み込み部 3903 は、カレントフラグ記憶部 3907 に記憶されたフラグの値に基づいて、命令格納部 3901 から仮想マシン命令又はデコードデータを読み出し、それぞれ、検索部 4405 又は実行部 3910 に出力する。つまり、デコードデータを読み込んだ場合には、検索部 4405 をバイパスして直接実行部 3910 にデコードデータを渡す。

#### 【0175】

カレントフラグ読出制御部 3912 は、デコード部 3902 から送られてくるデコードデータが分岐命令に相当するか監視し、分岐命令の場合には、その分岐命令の実行直後に、デコード部 3902 を制御することにより、その分岐先の命令ブロックのフラグ領域に格納されたフラグを読み出してカレントフラグ記憶部 3907 に格納させる。

（仮想マシンの動作）

以上のように構成された仮想マシン3900の動作について説明する。

【0176】

図54は、デコード部3902の動作を示すフローチャートである。

デコード部3902の命令読み込み部3903は、信号線Rを介して実行部3910から次の仮想マシン命令を読み込む旨の通知を受けると（ステップ8702、8703）、カレントフラグ記憶部3907に記憶されたフラグを読み出して、その内容を判断する（ステップ8704）。

【0177】

その結果、デコード命令列が存在しないと判断した場合には、命令読み込み部3903は、実施形態5の場合と同様の動作、即ち、PC3804に格納された仮想マシンコード領域上のアドレスに従って非分岐命令格納領域又は分岐命令格納領域に格納された仮想マシンコードを読み出し、検索部4405に渡す（ステップ8705、8706）。そして、検索部4405は、デコードテーブル4406を検索することでジャンプアドレスを特定し、デコードデータとして実行部3910に出力した後に（ステップ8707）、その旨を信号線Rで通知する（ステップ8711）。

【0178】

一方、上記カレントフラグによってデコード命令列が存在すると判断した場合には、命令読み込み部3903は、PC3804に格納された仮想マシンコード領域上のアドレスに一定のオフセットを加算することで実マシンコード領域8607a～8607d上のアドレスを算出した後に（ステップ8708）、そのアドレスに従ってデコードデータを読み出し（ステップ8709）、それを直接実行部3910に出力する（ステップ8710）。

【0179】

図55は、実行部3910の動作を示すフローチャートである。

本図と図79と比較して分かるように、基本的な流れは従来と同じである。つまり、PC3804やSP4412を初期化した後に（ステップ8802）、デコード部3902から送られてくるデコードデータに基づいてマイクロプログラム記憶部4411内のマイクロプログラムを実行する（ステップ8804～88

08)。

#### 【0180】

異なる点は、カレントフラグ記憶部3907についての処理（ステップ8803）が追加されていることである。つまり、実行部3910は、動作の開始にあたって、デコード命令列が存在しない旨のフラグを初期値としてカレントフラグ記憶部3907に格納しておく（ステップ8803）。

図56は、実行部3910が分岐命令を実行した場合のデコード部3902に対する制御を示すフローチャートである。本図と図42とを比較して分かるように、実行部3910が分岐命令を実行する場合においては、分岐先変換部3811は、分岐命令のオペランドを分岐先と命令ブロックの識別子セグメントとオフセットに変換し、それぞれPC3804の識別子セグメントレジスタ3804aとオフセットカウンタ3804bに格納するが（ステップ8902、8904）、この処理は実施形態5の場合と同じである。

#### 【0181】

異なる点は、カレントフラグ記憶部3907についての処理（ステップ8904）が追加されていることである。つまり、分岐先変換部3811によってPC3804が更新された後に（ステップ8902、8904）、カレントフラグ読出制御部3912は、命令読込み部3903を制御することによって、識別子セグメントレジスタ3804aに格納された識別子セグメントが示す命令ブロックのフラグ領域の値を読み出し、カレントフラグ記憶部3907に格納させる（ステップ8904）。これによって、新たな命令ブロックに分岐する度に、カレントフラグ記憶部3907の内容は更新され、これから実行する命令ブロックの実マシンコード領域にデコード命令列が格納されているか否かを示すフラグがカレントフラグ記憶部3907にセットされる。

#### 【0182】

以上のように、本実施形態に係る仮想マシン3900によれば、命令格納部3901には、実行対象となる仮想マシンプログラムが基本ブロックを単位とする命令ブロックに分割されて格納されており、さらに、各命令ブロックには仮想マシン命令だけでなく、それら仮想マシン命令に対応するデコードデータも配置さ

れる。そして、デコード部 3902 は、各命令ブロックのフラグ領域を参照することで、デコードデータが置かれた命令ブロックについては、単にそれらデコードデータを読み出して実行部 3910 に渡すだけで足り、検索部 4405 によるデコードテーブル 4406 の検索処理は省略される。これにより、実施形態 5 に係る仮想マシン 3800 と同等の効果が得られることに加え、予めデコード命令列が配置された命令ブロックについての実行時間は短縮される。

#### 【0183】

なお、本実施形態では、各命令ブロックにおける仮想マシンコード領域と実マシンコード領域とは、そのアドレスが一定のオフセットだけ離れているという位置関係にあったが、例えば、デコード命令列格納領域の先頭位置を特定するオフセットアドレスを各命令ブロック中に配置しておくことで、これら位置関係による制約を不要とすることができる。その場合には、新たな命令ブロックに分岐する度に、その命令ブロックのフラグと共に、そのオフセットアドレスを読み出すことで、カレントフラグに応じた仮想マシンコード領域及び実マシンコード領域それぞれに適したアドレスを PC 3804 にセットすることができる。

#### (実施形態 7)

次に、上記実施形態 6 に係る仮想マシンにおけるデコード命令列を動的に生成する実施形態 7 に係る仮想マシンについて説明する。

#### (仮想マシンの構成)

図 57 は、本実施形態に係る仮想マシン 4000 の構成を示す機能ブロック図である。本仮想マシン 4000 は、命令格納部 3901、デコード部 4002、実行部 3910 及びスタック 4420 から構成される。

#### 【0184】

本図と図 52 とを比較して分かるように、本仮想マシン 4000 は、実施形態 6 に係る仮想マシン 3900 とほぼ同一の構成要素を備える。相違する点は、デコード部 4002 にデコード命令列書込み部 4008 が追加されていること、その追加に伴うデコード部 4002 内部の接続変更である。以下、実施形態 6 に係る仮想マシン 3900 と相違する点を中心に説明する。

#### 【0185】

デコード命令列書込み部 4008 は、本仮想マシン 4000 による実行制御がデコード命令列を持たない命令ブロックに分岐した場合に、その命令ブロックに対する実行を中断させ、一旦、その命令ブロックに置かれている仮想マシンプログラム全体を命令読込み部 3903 及び検索部 4405 によってデコード命令列に変換させた後に、そのデコード命令列を当該命令ブロックのデコード命令列格納領域に書き込む。そして、書き込まれたデコード命令列に対して、命令読込み部 3903 による読込みと実行部 3910 による実行を再開させる。

【0186】

従って、結果的に、命令格納部 3901 から命令読込み部 3903 によって読み出されたデコードデータだけがそのまま実行部 3910 に渡され、検索部 4405 がデコードテーブル 4406 を検索することによって得たデコードデータは直接には実行部 3910 に渡されない。このことは、実施形態 6 の場合と相違し、検索部 4405 からのデコードデータは、実行部 3910 ではなく、デコード命令列書込み部 4008 に送られていることに対応する。

(仮想マシンの動作)

以上のように構成された仮想マシン 4000 の動作について説明する。

【0187】

図 58 は、本仮想マシン 4000 が分岐命令を実行した場合の本仮想マシン 4000 における特徴的な動作、即ち、デコード命令列書込み部 4008、カレントフラグ読出制御部 3912 及び分岐先変換部 3811 における動作を示すフローチャートである。新たな命令ブロックへの分岐に際して、分岐先変換部 3811 が PC 3804 の値を更新し（ステップ 9102、9103）、カレントフラグ読出制御部 3912 がカレントフラグ記憶部 3907 の内容を更新する点は、図 56 に示された実施形態 6 の手順と同じである。異なる点は、その後に、必要に応じて、デコード命令列書込み部 4008 によるデコード命令列の生成と命令格納部 3901 への書き込みが行われることである（ステップ 9105～9111）。

【0188】

具体的には、デコード命令列書込み部 4008 は、命令読込み部 3903 によ

って読み出されたフラグを受け取って参照することによって、その命令ブロックには既にデコード命令列が格納されているか否か判断する（ステップ9105）。

その結果、デコード命令列が格納されていると判断した場合には、デコード命令列書込み部4008は、特になんの処理も行わない（ステップ9112）。このときには、その命令ブロックに格納されているデコード命令列が順次読み出されて実行部3910によって直接に実行されることになる。

#### 【0189】

一方、デコード命令列が格納されていないと判断した場合には、デコード命令列書込み部4008は、ポインタdPCをインクリメントしながら（ステップ9106～9111）、その命令ブロックに置かれている仮想マシンコードを命令読み込み部3903によって順次読み出させ（ステップ9108、9109）、読み出された仮想マシンコードを検索部4405によって必要なジャンプアドレス等のデコードデータに変換させた後に当該命令ブロックのデコード命令列格納領域に書き込んでいく（ステップ9110）。

#### 【0190】

そして、当該命令ブロックの全ての仮想マシンコードについてデコードデータへの変換と書き込みを終えると（ステップ9107）、デコード命令列が存在する旨のフラグをカレントフラグ記憶部3907及び当該命令ブロックのフラグ領域に書き込むことで、処理を終える（ステップ9112）。これによって、当該命令ブロックのデコード命令列に対する命令読み込み部3903による読み込みと実行部3910による実行が再開される。

#### 【0191】

図59は、図58におけるステップ9110、即ち、仮想マシンコードからデコードデータへの変換と命令格納部3901への格納処理の詳細を示すフローチャートである。本図と図77とを比較して分かるように、従来の検索部4405による処理に、デコード命令列書込み部4008による処理、つまり、デコードテーブル4406の検索によって得られたジャンプアドレスddや仮想マシン命令のオペランドをデコードデータとして命令格納部3901のデコード命令列格納



領域に書き込む処理（ステップ 9209、9213）が追加されていることがわかる。

#### 【0192】

図 60 は、実行部 3910 から見たデコード部 4002 の動作を示すフローチャートである。命令読み込み部 3903 は、実行部 3910 との関係においては、命令格納部 3901 の実マシンコード領域からデコードデータを読み出した場合にだけ、そのデコードデータを実行部 3910 に渡すので、デコードデータ専用の読み出し部として機能していると言える。

#### 【0193】

以上のように、本実施形態に係る仮想マシン 4000 によれば、デコード命令列を持たない命令ブロックに分岐した場合に、一旦、その命令ブロックの仮想マシンプログラムはデコードデータに変換され、命令格納部 3901 に書き戻された後に、そのデコードデータが直接に実行される。従って、再び、その命令ブロックを実行する場合には、デコードデータを読み出して直接に実行することができるので、解読のための時間、即ち、検索部 4405 によるデコードテーブル 4406 の検索に伴う時間が削減される。特に、ループ処理のように、同一の命令ブロックを繰り返して実行する場合における実行速度は飛躍的に向上される。

#### （実施形態 8）

次に、上記実施形態 7 に係る仮想マシンを圧縮符号化された仮想マシンプログラムに対応させた実施形態 8 に係る仮想マシンについて説明する。

#### （仮想マシンの構成）

図 61 は、本実施形態に係る仮想マシン 4100 の構成を示す機能ブロック図である。本仮想マシン 4100 は、命令格納部 4101、デコード部 4102、実行部 3910 及びスタック 4420 から構成される。

#### 【0194】

本図と図 57 とを比較して分かるように、本仮想マシン 4100 は、実施形態 7 に係る仮想マシン 4000 とほぼ同一の構成要素を備える。相違する点は、命令格納部 4101 に格納されている仮想マシンプログラムのコード形態、命令格納部 4101 に展開情報格納領域 4157a～4157d が設けられていること

、デコード部4102の命令読み込み部4103に仮想マシン命令展開部4103aが追加されていることである。以下、実施形態7に係る仮想マシン4000と相違する点を中心に説明する。

#### 【0195】

命令格納部4101の非分岐命令格納領域4154a～4154d及び分岐命令格納領域4155a～4155d（これら領域を併せて、本実施形態では、「圧縮仮想マシンコード領域」と呼ぶ。）は、圧縮符号化された仮想マシン命令を予め保持している。また、命令格納部4101の展開情報格納領域4157a～4157dは、その命令ブロックに格納された圧縮符号化された仮想マシン命令を伸長復号化するための復号化テーブルを予め保持している。

#### 【0196】

図62（a）は、その復号化テーブルの例を示す。圧縮されたビット列と、対応する仮想マシン命令との組が示されている。

図62（b）は、図62（a）に示された復号化テーブルにおける符号の規則性を示す図である。ここでは、ハフマン符号化に準じたビット圧縮方法で、オペランドを含む仮想マシン命令の単位でビット列に圧縮していくものとする。例えば、ビット列“000”は仮想マシン命令“Push [0]”を意味し、ビット列“01010”は仮想マシン命令“Push 10”を意味する。

#### 【0197】

図63（a）～（c）は、命令格納部4101に格納されている仮想マシンプログラムの格納状態の例を示す図であり、図97に示されたサンプル仮想マシンプログラムが格納されている場合に相当する。各命令ブロック格納領域4152a～4152cにおいて、非分岐命令格納領域4154a～4154c及び分岐命令格納領域4155a～4155cからなる各圧縮仮想マシンコード領域4164a～4164cには、各命令ブロックの仮想マシンプログラムを圧縮符号化して直列に接続することによって得られるビット列（以下、「圧縮ビット列」と呼ぶ。）が予め格納され、各展開情報格納領域4157aには、当該ビット列を伸長復号化するための復号化テーブルが格納されている。なお、図63（b）には、デコード命令列を持たない命令ブロック格納領域4152bが示され、図6

3(c)には、デコード命令列を持つ命令ブロック格納領域4152cが示されている。命令読み込み部4103は、実施形態7の命令読み込み部3903が備える機能、即ち、命令格納部4101の圧縮仮想マシンコード領域4164a～4164dから圧縮ビット列を読み出し、デコード命令列格納領域4156a～4156dからデコード命令列を読み出すことに加え、さらに、仮想マシン命令展開部4103aを備える。

#### 【0198】

仮想マシン命令展開部4103aは、命令読み込み部4103が命令格納部4101の圧縮仮想マシンコード領域4164aから圧縮ビット列を1ビットずつ読み出す度にその命令ブロックの展開情報格納領域4157a～4157dに格納された復号化テーブルを参照することで、対応する仮想マシン命令を特定し、特定した仮想マシン命令を検索部4405に渡すという伸長復号化を繰り返す。

#### (仮想マシンの動作)

以上のように構成された仮想マシン4100の動作について説明する。

#### 【0199】

上述したように、本仮想マシン4100は、実施形態7に係る仮想マシン4000が備える機能を全て含むので、圧縮ビット列の復号処理を除く全体的な処理は実施形態7と同じであり、図58に示されるフローチャートの通りである。

つまり、本仮想マシン4100は、実施形態7に係る仮想マシン4000と同様に、デコード命令列を持たない命令ブロックに分岐した場合には、命令読み込み部4103及び検索部4405によって、一旦、その命令ブロックの仮想マシンプログラムをデコードデータに変換し、デコード命令列書き込み部4008によって命令格納部3901に書き戻した後に、命令読み込み部4103及び実行部3910によってデコード命令列を直接に実行する。

#### 【0200】

ところが、本実施形態の仮想マシン4100は、圧縮符号化された仮想マシン命令を読み出すので、図58におけるステップ9109とステップ9110での詳細な処理が実施形態7と異なる。圧縮ビット列に適した読み出しを行う必要があること、及び、その復号化処理を追加する必要があるためである。

図64は、図58におけるステップ9109及びステップ9110の詳細を示すフローチャートであり、本仮想マシン4100のデコード部4102の動作を示す。ここで、図64におけるステップ9602、ステップ9603～9616は、それぞれ、図58におけるステップ9109、ステップ9110に相当する。

#### 【0201】

本図と実施形態7における図59とを比較して分かるように、相違する点は、本実施形態では、直接に仮想マシンコードを読み出すことに替えて、圧縮ビット列の読み出しと復号化を行っていることと（ステップ9602）、その復号化において必要なオペランド（配列op[i]）が同時に得られるので、命令格納部4101からオペランドだけを読み出すことなくそのオペランド（配列op[i]）をデコード命令列格納領域4156a～4156dに書き込んでいること（ステップ9613）である。

#### 【0202】

図65は、図64におけるステップ9602の詳細を示すフローチャートである。命令読み込み部4103は、まず、圧縮ビット列の一時的な記憶域（変数bits）を確保した後に（ステップ9702）、PC3804の値に従って、デコード命令列を持たない命令ブロック格納領域4152a～4152dの圧縮仮想マシンコード領域4164a～4164dから1ビット分の圧縮符号を読み出し（ステップ9803）、それと既に読み出した圧縮符号（変数bits）とを連結する（ステップ9704）。

#### 【0203】

そして、仮想マシン命令展開部4103aは、PC3804の値に一定のオフセットを加算することによって得たアドレスから始まる展開情報格納領域4157a～4157dの復号化テーブル、即ち、読み出された圧縮ビット列が置かれていた命令ブロックの復号化テーブルに登録されている圧縮ビット列それぞれと上記ステップ9704で得られた圧縮符号（変数bits）とを順次に比較することで、一致する仮想マシン命令を特定する（ステップ9705）。これら読み出し（ステップ9703）と検索（ステップ9705）は、一致する仮想マシン命令

が見つかるまで繰り返す（ステップ9706）。

#### 【0204】

一致する仮想マシン命令が見つかり、仮想マシン命令展開部4103aは、その仮想マシン命令を展開情報格納領域4157a～4157dから読み出し（ステップ9707）、その仮想マシン命令を、又は、その仮想マシン命令にオペランドが含まれる場合にはその仮想マシン命令中のオペコードとオペランド（配列op[]）とを分離した後に検索部4405に出力する（ステップ9708、9709）。その後、図64のステップ9603～9614に示されるように、検索部4405によって対応するデコードデータに変換され、デコード命令列書込み部4008によって、必要なオペランド配列op[]と共に当該命令ブロックの実マシンコード領域に書き込まれる。

#### 【0205】

以上のように、本実施形態に係る仮想マシン4100によれば、命令格納部4101の各命令ブロックには圧縮符号化された仮想マシンプログラムが置かれ、本仮想マシン4100がデコード命令列を持たない命令ブロックに分岐した場合には、一旦、その命令ブロックに置かれた圧縮符号化された仮想マシンプログラムが伸長復号化された後に、デコードデータに変換されて命令格納部3901に書き戻され、そのデコードデータが直接に実行される。

#### 【0206】

従って、本仮想マシン4100によれば、圧縮ビット列は、常に命令ブロックの先頭、即ち、完全な1個の仮想マシン命令の先頭から復号されることが保証される。これによって、分岐命令の実行に伴って1個の仮想マシン命令の圧縮ビット列の途中から誤って復号化してしまうことが確実に回避され、圧縮符号化された仮想マシンプログラムを正しく実行することが可能な仮想マシンが実現される。

#### 【0207】

なお、本実施形態では、命令格納部4101の各命令ブロック格納領域4152a～4152dにはデコード命令列格納領域4156a～4156dが設けら

れていたが、圧縮ビット列の途中から誤って復号化されるとい従来の問題点を回避するだけならば、これらデコード命令列格納領域4156a～4156dを設けなくてもよい。

#### 【0208】

つまり、本実施形態に係る仮想マシン4100は、デコード命令列格納領域4156a～4156dを備えた実施形態7に係る仮想マシン4000を、圧縮符号化された仮想マシンプログラムを解読実行するように対応させたものに相当するが、デコード命令列格納領域を備えない実施形態5に係る仮想マシン3800を、圧縮符号化された仮想マシンプログラムを解読実行するように対応させたものであってもよい。いずれの場合であっても、圧縮符号化された仮想マシンプログラムは基本ブロックを単位として格納され、全ての分岐命令が指定する分岐先は基本ブロックの先頭であることが保証されているので、圧縮ビット列の途中から誤って復号されてしまうことは回避されるからである。

#### 【0209】

なお、本実施形態では、圧縮符号化の手法としてハフマン符号化を用いたが、LZ法等の他の圧縮手法を用いても、なんら問題がないのは言うまでもない。

#### (実施形態9)

次に、対象とする実マシン固有のジャンプ先についての境界制限を満足する実マシンコードを高速に生成する実施形態9に係るその場式コンパイラについて説明する。

#### (コンパイラシステムの構成)

図66は、本実施形態に係るその場式コンパイラ4300を含むコンパイラシステム全体の構成を示す機能ブロック図である。つまり、本図には、その場式コンパイラ4300だけでなく、その場式コンパイラ4300への入力に必要な情報を生成する仮想マシンコンパイラ4320も併せて示されている。

#### 【0210】

仮想マシンコンパイラ4320は、C言語等の高級言語で記述されたソースプログラム4310を入力とし、特定の仮想マシンを対象とする仮想マシンコードを生成して経路D1に出力する一般的なコンパイラが備える言語変換機能に加えて

、その場式コンパイラ4300で必要とされる特有の情報（ブロック先頭情報）を生成し経路D2に出力するブロック先頭情報生成部4321aをも備える。

#### 【0211】

ブロック先頭情報生成部4321aは、一般的なコンパイラが備える出力部4321、つまり、構文解析や中間コードへの変換等を経て得られた最終的な仮想マシンコードを外部に順次出力する出力部4321に付加された機能であり、出力部4321から経路D1に出力される各仮想マシンコードごとに、その仮想マシンコードが基本ブロックの先頭とすべきものか否か判断し、その結果を示すブロック先頭情報を経路D2に出力する。

#### 【0212】

その場式コンパイラ4300は、上記仮想マシンコンパイラ4320が生成した仮想マシンコード及びブロック先頭情報を入力とし、実マシン命令の分岐先がアドレス空間の2ワード境界に制限された実マシンを対象とする実マシン命令列4311に変換するコンパイラであり、実マシン命令変換部4301、分岐位置補正部4302及び実マシンアドレス保持部4303とから構成される。

#### 【0213】

実マシン命令変換部4301は、仮想マシンコンパイラ4320から経路D1を経て出力された仮想マシンコードがオペコードの場合には内部の変換テーブルに基づいて対応する実マシンコードに変換し、一方、その仮想マシンコードがオペランドの場合にはそのまま、分岐位置補正部4302に出力する。そのとき、実マシン命令変換部4301は、実マシンアドレス保持部4303が保持する実マシンアドレスPCを読み出して、上記実マシンコードと共に分岐位置補正部4302に出力した後に、その実マシンアドレスPCをインクリメントしておく。

#### 【0214】

実マシンアドレス保持部4303は、実マシン命令変換部4301が次に生成する実マシンコードを配置すべき実マシン空間での相対アドレスPCを保持する。

分岐位置補正部4302は、実マシン命令変換部4301から送られてきた実マシンアドレスPCと仮想マシンコンパイラ4320から経路D2を経て出力されたブロック先頭情報とから、基本ブロックの先頭に位置する実マシン命令が奇数番

地、即ち、2ワード境界を違反する位置に配置されることとなるか否かを判断し、違反することとなる場合には、1ワード分のダミー命令、即ち、無動作の実マシン命令"Nop"を前置させた後に、実マシン命令変換部4301から送られてきた実マシンコードを実マシン命令列4311として外部に出力する。これは、処理内容を変更することなく、基本ブロックの実質的な先頭を2ワード境界に配置させるためである。

(コンパイラシステムの動作)

以上のように構成されたコンパイラシステムの動作について、一般的なコンパイラと異なる点を中心に説明する。

【0215】

図67は、仮想マシンコンパイラ4320のブロック先頭情報生成部4321aの動作を示すフローチャートである。図47に示された実施形態5に係る仮想マシンコンパイラの動作と基本的な流れは同じである。

まず、ブロック先頭情報生成部4321aは、出力部4321が出力しようとする仮想マシンコードVCそれぞれについて、基本ブロックの先頭とすべきものかどうかを判定する(ステップ10003、10004)。その結果、基本ブロックの先頭とすべき仮想マシンコードと判定した場合にはその旨を示すブロック先頭情報"T"を(ステップ10006)、一方、先頭とすべきでないと判定した場合にはその旨を示すブロック先頭情報"N"を(ステップ10005)、その仮想マシンコードVCと共に、それぞれ、経路D1及びD2に出力する(ステップ10007)。

【0216】

図68は、実マシン命令変換部4301、分岐位置補正部4302及び実マシンアドレス保持部4303の動作を示すフローチャートである。まず、実マシンアドレス保持部4303は、実マシンアドレスPCを初期化しておく(ステップ10102)。

実マシン命令変換部4301は、ブロック先頭情報生成部4321aから出力された仮想マシンコードVCを受け取ると(ステップ10103、10104)、必要に応じて対応する実マシンコードに変換し、それを実マシンアドレス保持部



4303から読み出した実マシンアドレスPCと共に分岐位置補正部4302に渡した後に、実マシンアドレスPCをインクリメントしておく（ステップ10105）。

#### 【0217】

次に、分岐位置補正部4302は、上記仮想マシンコードVCに対応するブロック先頭情報BIをブロック先頭情報生成部4321aから受け取ると（ステップ10106）、実マシン命令変換部4301から送られてきた実マシンコードを外部に出力するに際して、上記境界違反を生じることとなるか否か判断する（ステップ1017、10108）。具体的には、ブロック先頭情報生成部4321aから受け取ったブロック先頭情報BIが基本ブロックの先頭である旨“T”を示し、かつ、実マシン命令変換部4301から受け取った実マシンアドレスPCが2ワード境界にないか否かを判断する（ステップ10107、10108）。

#### 【0218】

その結果、基本ブロックの先頭であり、かつ、2ワード境界にないと判断した場合には、上記境界違反を回避するために、実マシン命令“Nop”を生成し出力した後に上記実マシン命令を実マシン命令列4311として出力し（ステップ10109、10110）、そうでない場合には、単に上記実マシン命令を実マシン命令列4311として出力する（ステップ10110）。なお、分岐位置補正部4302は、実マシン命令“Nop”を生成した場合には（ステップ10110）、そのサイズ分だけ実マシンアドレス保持部4303の実マシンアドレスPCを更新しておく。

#### 【0219】

以上の処理（ステップ10104～10110）は、ブロック先頭情報生成部4321aから仮想マシンコードが送られてくる限り、繰り返す（ステップ10103、10111）。

図69は、図97に示されたサンプル仮想マシン命令列が本その場式コンパイラ4300に入力された場合において、ブロック先頭情報生成部4321aによって生成されるブロック先頭情報、分岐位置補正部4302によって生成される実マシン命令“Nop”のタイミング、その他の関連情報を示すテーブルである。本

図から分かるように、仮想マシンアドレス0番、アドレス8番、アドレス15番及びアドレス31番の仮想マシン命令は、基本ブロックの先頭となるので、その旨のブロック先頭情報”T”が生成される。

【0220】

また、仮想マシンアドレス15番においては、分岐位置補正部4302は、ブロック先頭情報生成部4321aからブロック先頭情報”T”を受け取り、実マシン命令変換部4301から奇数（35）の実マシンアドレスPCを受け取るので、仮想マシン命令”Push [1]”に対応する実マシン命令を出力する前に、実マシン命令”Nop”を出力する。これによって、ブロックの先頭命令が奇数番地に置かれることが回避される。

【0221】

以上のように、本実施形態に係るその場式コンパイラ4300は、従来において必要とされた分岐命令の分岐先についての解析という複雑な処理を行うことなく、ジャンプ先についての境界違反を起こすことがない実マシンプログラムを生成することができた。これは、仮想マシンコンパイラ4320において、ブロック先頭情報生成部4321aが基本ブロックの検出を行い、その検出結果であるブロック先頭情報をその場式コンパイラ4300に伝達しているからである。

【0222】

従って、本発明に係るその場式コンパイラ4300は、従来のその場式コンパイラと比較し、ブロック先頭情報に基づく実マシン命令”Nop”の追加挿入という簡単な処理を追加するだけで、境界違反の問題を解消している。これによって、ジャンプ命令のジャンプ先についての境界制限を違反することがない適正な実マシンコードが短時間で生成されるその場式コンパイラが実現される。

【0223】

なお、本実施形態に係るブロック先頭情報生成部4321aは、仮想マシンコンパイラ4320の出力部4321に付随して備えられたが、これに代えて、一般的なコンパイラが備える基本ブロックへの分割処理を利用してもよい。つまり、一般的なコンパイラは、通常、最適化等の過程において、基本ブロックへの分割処理を行うので、その分割処理の過程で得られたブロック先頭情報を外部（そ

の場式コンパイラ 4300) に出力させることで、容易にブロック先頭情報生成部 4321a を実現することができる。

【0224】

また、本実施形態では、ジャンプ先となる実マシン命令の配置についてのみアラメント処理が行われたが、同様な方法によって、遅延分岐や打ち消し分岐のための配置を行うことができるのは言うまでもない。

以上、本発明に係る仮想マシン、仮想マシンコンパイラ及びその場式コンパイラについて、実施形態 1～9 に基づいて説明したが、本発明は、これら実施形態に限定されるものではない。各実施形態に示された特徴的な構成要素を組み合わせたり分離したりすることで、容易に本発明のバリエーションを考案することができる。

【0225】

例えば、実施形態 1 と実施形態 5 とを組み合わせることで、仮想マシンプログラムが基本ブロック単位で分割され、かつ、対応する先行命令情報と共に命令格納部に格納しておくことができる。これによって、真のデータ依存関係を解消し、かつ、その場式コンパイラによるアドレス解決処理を単純化する高速な仮想マシンが実現される。

【0226】

また、実施形態 2 と実施形態 8 とを組み合わせることで、適度な頻度で割込み処理が行われ、かつ、適正に復号されることが保証された圧縮ビット列を実行する割込み機能付き仮想マシンが実現される。

また、実施形態 1 における先行命令情報と仮想マシン命令、実施形態 9 におけるブロック先頭情報と仮想マシン命令とは、それぞれ独立分離された構造であったが、例えば、図 70 に示されるように、本発明に係る仮想マシンが実行する仮想マシン命令として、これら先行命令情報、ブロック先頭情報が埋め込まれた命令フォーマットを有する拡張仮想マシン命令を定義してもよい。その場合には、拡張仮想マシン命令を単位として命令格納部等から読み込んだ後に、定型的に分割することで、先行命令情報、ブロック先頭情報、仮想マシン命令のオペコード及びオペランドを区別して得ることができる。

## 【0227】

また、実施形態5～8では、各命令ブロック格納部はユニークな識別子を保持していたが、各命令ブロックを一定のルールに従って命令格納部に配置する等により、それらを個別に識別できるならば、各命令ブロック格納部に識別子を保持させる必要はない。

さらに、本発明に係る仮想マシン、仮想マシンコンパイラ及びその場式コンパイラは、いずれも、汎用のコンピュータによって実行されるプログラムとして実現することができるので、これら発明に係るプログラムをCD-ROM等の記録媒体に収納したり、伝送路を介して通信したりすることによって、流通させることができるのは言うまでもない。

## 【0228】

## 【発明の効果】

以上の説明から明らかなように、本発明に係る仮想マシンは、実マシンによる制御の下で仮想マシン命令を実行するスタック型仮想マシンであって、データを後入れ先出し方式で一時的に記憶するためのスタック手段と、実行対象となる仮想マシン命令列と、それら仮想マシン命令それぞれに対応づけられた情報であって、対応づけられた仮想マシン命令に後続する仮想マシン命令が実行された場合の前記スタック手段でのデータの格納状態の変化を示す先行命令情報とを記憶する命令記憶手段と、前記命令記憶手段から次に実行すべき仮想マシン命令及び対応する先行命令情報を読み出す読み出し手段と、読み出された仮想マシン命令と先行命令情報との組み合わせに対応する演算処理を特定し実行する解読実行手段とを備えることを特徴とする。

## 【0229】

これによって、命令記憶手段には、仮想マシン命令だけでなく先行命令情報をも格納されているので、解読実行部は、仮想マシン命令の演算処理と共に、後続する仮想マシン命令のためのスタック操作を先行して処理しておくことができるので、例えば、スーパースカラマシンにおいて特に生じ易いパイプラインハザードのマシンサイクルにおいてその先行処理を実行しておくことにより、真のデータ依存関係の影響が吸収され、実行速度が高速化されるという効果がある。

## 【0230】

ここで、前記解説実行手段は、対象とする全ての種類の仮想マシン命令と全ての種類の先行命令情報との組み合わせそれぞれに対応する実マシン命令列を記憶する実マシン命令列記憶部と、前記読み出し手段によって読み出された仮想マシン命令と先行命令情報との組み合わせに対応する実マシン命令列を前記実マシン命令列記憶部に記憶された実マシン命令列から特定する特定部と、特定された実マシン命令列を実行する実行部とを有することとすることもできる。

## 【0231】

これによって、データ依存関係を吸収するための上記先行処理を各仮想マシン命令に対応する実マシン命令列に組み入れておくことができる。

また、前記先行命令情報は、対応する仮想マシン命令に後続する仮想マシン命令が実行された場合の前記スタック手段に格納されるデータ数の増減を示し、前記実マシン命令列記憶部に記憶された実マシン命令列には、対応する先行命令情報に基づいて前記スタック手段を先行処理する実マシン命令が含まれていることとすることもできる。

## 【0232】

これによって、当該仮想マシン命令の実行に伴うスタック段数の増減と、後続する仮想マシン命令の実行に伴うスタック段数の増減とが、それぞれ打ち消し合う操作である場合においては、無駄なスタック操作が回避され、実行速度が向上される。

また、前記実マシン命令列記憶部に記憶された実マシン命令列は、前記スタック手段において最後に格納されたデータと最後から2番目に格納されたデータそれぞれを記憶する領域が前記実マシンが備える2つのレジスタに割り当てられていることを前提とする内容であることとすることもできる。

## 【0233】

これによって、スタック型仮想マシンにおいて頻繁に行われるスタックへのデータの出し入れは、実マシンの内部レジスタへの読み書き動作となるので、パイプラインハザードを埋めるのに好適な先行処理となり、実行効率の向上に適したアーキテクチャとなる。

また、前記命令記憶手段は、前記仮想マシン命令列を記憶する第1領域と、その第1領域における各仮想マシン命令の記憶位置に関連付けられた記憶位置に前記先行命令情報を記憶する第2領域とからなり、前記読み出し手段は、前記第1領域と前記第2領域それぞれの対応する記憶領域から前記仮想マシン命令と先行命令情報との組を読み出すこととすることもできる。

## 【0234】

これによって、仮想マシン命令列は、先行命令情報と分離されて格納されるので、従来の仮想マシン命令列と同一のフォーマットとあり、従来の仮想マシンとの命令フォーマットの互換性が維持される。

また、前記命令記憶手段は、前記仮想マシン命令列と前記先行命令情報とを、対応する1つの仮想マシン命令と1つの先行命令情報との組を1つの拡張仮想マシン命令とする拡張仮想マシン命令列として記憶し、前記読み出し手段は、前記命令記憶手段から拡張仮想マシン命令を読み出し、前記解読実行手段は、読み出された拡張仮想マシン命令に対応する演算処理を特定し実行することとすることもできる。

## 【0235】

これによって、拡張仮想マシン命令を新たな1個の仮想マシン命令と扱うことで、先行命令情報だけを特別に処理したり記憶させたりする処理が不要となり、全体として、従来のコンピュータのアーキテクチャに近い仮想マシンが実現される。

また、本発明に係るコンパイラは、スタック型仮想マシンを対象とするコンパイラであって、ソースプログラムを前記仮想マシンが実行する仮想マシン命令列に変換する命令列変換手段と、変換によって得られた仮想マシン命令列を構成する仮想マシン命令それぞれについて、後続する仮想マシン命令が前記仮想マシンによって実行された場合の前記スタック手段でのデータの格納状態の変化を示す先行命令情報を生成する先行命令情報生成手段と、生成された先行命令情報と対応する仮想マシン命令とを関連付けて出力する関連付け手段とを備えることを特徴とする。

## 【0236】

これによって、仮想マシン命令だけでなく先行命令情報をも生成されるので、真のデータ依存関係の影響を吸収することで高速実行が可能な仮想マシンを対象とする仮想マシンコンパイラが実現される。

また、本発明に係る仮想マシンは、実マシンによる制御の下で仮想マシン命令を実行する仮想マシンであって、実行対象となる仮想マシン命令列を記憶する命令記憶手段と、前記命令記憶手段から次に実行すべき仮想マシン命令を読み出す読み出し手段と、読み出された仮想マシン命令に対応する演算処理を特定し実行する解読実行手段とを備え、前記解読実行手段は、読み出された仮想マシン命令が実行制御の流れを分岐させる分岐命令であるか否かを判定する分岐命令判定部と、分岐命令であると判定される度に、その分岐命令の実行に加えて、この仮想マシンへの割込み要求の発生の有無の検出と、発生している場合の割込み処理とを実行する割込み処理部とを有することを特徴とする。

#### 【0237】

これによって、1個の分岐命令が実行される度に、1回の割込み処理が実行されるので、多くの仮想マシンプログラムにおいては適度な頻度で割込み処理が繰り返されることとなり、頻繁な割込み処理が行われることによる実行速度の低下が回避されるという効果がある。

ここで、前記解読実行手段はさらに、対象とする全ての種類の仮想マシン命令に対応する実マシン命令列とこの仮想マシンへの割込み要求に対処するための実マシン命令列を記憶する実マシン命令列記憶部と、指定された実マシン命令列を実行する実行部とを備え、前記割込み処理部は、前記分岐命令判定部によって分岐命令であると判定される度に、前記実行部に、前記割込み要求に対処するための実マシン命令列を実行させた後に前記分岐命令に対応する実マシン命令列を実行させるとすることもできる。

#### 【0238】

これによって、仮想マシン命令ではなく、デコードデータによって、追加処理すべき割込み処理を指定することができるので、簡易な構成によって割込み処理を行う仮想マシンが実現される。

また、本発明に係る仮想マシンは、実マシンによる制御の下で仮想マシン命令

を実行する仮想マシンであって、実行対象となる仮想マシン命令列を記憶する命令記憶手段と、前記命令記憶手段から次に実行すべき仮想マシン命令を読み出す読み出し手段と、読み出された仮想マシン命令に対応する演算処理を特定し実行する解読実行手段とを備え、前記解読実行手段は、読み出された仮想マシン命令が一定個数の仮想マシン命令の集まりをブロックとした場合の各ブロックを代表する仮想マシン命令に相当するか否かを判定するブロック判定部を備え、各ブロックを代表する仮想マシン命令であると判定される度に、その仮想マシン命令の実行に加えて、この仮想マシンへの割込み要求の発生の有無の検出と、発生している場合の割込み処理とを実行する割込み処理部とを有することを特徴とする。

## 【0239】

これによって、一定数の仮想マシン命令が実行される度に、1回の割込み処理が実行されるので、割込み処理を発生させる頻度についての制御が可能となり、頻繁な割込み処理が行われることによる実行速度の低下が回避される。

ここで、前記解読実行手段はさらに、対象とする全ての種類の仮想マシン命令に対応する実マシン命令列とこの仮想マシンへの割込み要求に対処するための実マシン命令列を記憶する実マシン命令列記憶部と、指定された実マシン命令列を実行する実行部とを備え、前記ブロック判定部は、仮想マシン命令が読み出される度に、それまでに読み出された仮想マシン命令の総数が一定値の倍数に一致するか否かを比較し、一致する場合に、前記仮想マシン命令は各ブロックを代表する仮想マシン命令に相当すると判定し、前記割込み処理部は、前記分岐命令判定部によって各ブロックを代表する仮想マシン命令であると判定される度に、前記実行部に、前記割込み要求に対処するための実マシン命令列を実行させた後に前記仮想マシン命令に対応する実マシン命令列を実行させるとすることもできる。

## 【0240】

これによって、仮想マシン命令ではなく、デコードデータによって、追加処理すべき割込み処理を指定することができるので、簡易な構成によって割込み処理を行う仮想マシンが実現される。

また、本発明に係る仮想マシンは、実マシンによる制御の下で仮想マシン命令を実行する仮想マシンであって、実マシン命令からなる複数のサブプログラムを



記憶する実マシンプログラム記憶手段と、実行対象となる仮想マシン命令列を記憶する第1領域と、前記実マシンプログラム記憶手段における各サブプログラムへのポインタを記憶する第2領域とを含む命令記憶手段と、前記命令記憶手段の第1領域から次に実行すべき仮想マシン命令を読み出す読み出し手段と、読み出された仮想マシン命令に対応する演算処理を特定し実行する解読実行手段とを備え、前記解読実行手段は、読み出された仮想マシン命令が前記第2領域に実行制御を移す命令であるか否かを判定する領域判定部と、前記第2領域に実行制御を移す命令であると判定された場合に、その移動先の第2領域の箇所に格納された前記ポインタが示す前記サブプログラムを実行するアドレス変換実行部とを有することを特徴とする。

#### 【0241】

これによって、仮想マシン命令に代えて実マシン命令を実行させるか否かは、仮想マシンのメモリマップにおける領域によってのみ決定されるので、関数ごとに仮想マシン関数として実行させたり実マシン関数に置き換えて実行させたりする切り替え設定が容易となり、アーキテクチャの異なる複数の実マシンを対象として容易にネイティブコード化をしておくことが可能な仮想マシンが実現される。

#### 【0242】

ここで、前記命令記憶手段における第1領域と第2領域とは、一定のアドレス値を境界とする隣接した記憶位置に設けられ、前記領域判定部は、読み出された仮想マシン命令がサブプログラムを呼び出す命令である場合に、その呼び出し先アドレスと前記境界との大小関係を比較することによって前記判定を行うとすることもできる。

#### 【0243】

これによって、上記境界の位置を変更するだけで、仮想マシン関数への呼び出しをそのまま実行させたり実マシン関数に置き換えて実行させたりする制御が可能となるので、複数の異なるアーキテクチャの実マシンを対象とする場合であっても、そのために全体としての実行速度を低下させることなく個々の実マシンの環境に応じた実行速度の高速化が可能な仮想マシンが実現される。

## 【0244】

また、本発明に係る仮想マシンは、実マシンによる制御の下で仮想マシン命令を実行する仮想マシンであって、実行対象となる仮想マシン命令列を記憶する命令記憶手段と、前記命令記憶手段から次に実行すべき仮想マシン命令を読み出す読み出し手段と、読み出された仮想マシン命令に対応する演算処理を特定し実行する解読実行手段とを備え、前記命令記憶手段は、前記仮想マシン命令列を構成する基本ブロックそれぞれに対応する命令ブロックの集まりからなり、各命令ブロックは、その命令ブロックの前記命令記憶手段における先頭位置を特定するための識別子を格納した識別子領域と、対応する基本ブロックの非分岐命令だけを格納した非分岐命令領域と、対応する基本ブロックの分岐命令を格納した分岐命令領域とを含み、前記分岐命令領域に格納された分岐命令は、その分岐先が前記識別子によって指定され、前記解読実行手段は、読み出された仮想マシン命令が前記分岐命令である場合には、その分岐先として指定されている識別子に対応する命令ブロックの非分岐命令領域の先頭に実行制御を分岐させることを特徴とする。

## 【0245】

これによって、各命令ブロックへの入口は先頭の1箇所だけとなることが保証されるので、その場式コンパイラでの分岐先命令のアドレス解析処理は単純化され、コンパイル時間が短縮される。また命令キャッシングを命令ブロック単位で行うことで、キャッシュ境界に伴う判定処理も簡略化され、仮想マシンにキャッシュを導入した場合の実行効率の低下が従来よりも抑えられるという効果がある。

## 【0246】

ここで、前記解読実行手段は、次に読み出すべき仮想マシン命令が属する命令ブロックの識別子を格納する識別子レジスタと、その仮想マシン命令のその命令ブロックにおける相対的な記憶位置を示すオフセットを格納するオフセットカウンタとからなるプログラムカウンタを備え、前記読み出し手段は、前記プログラムカウンタに格納された識別子とオフセットに基づいて前記仮想マシン命令を読み出し、前記解読実行手段は、読み出された仮想マシン命令が前記分岐命令であ

る場合には、その分岐先として指定されている識別子を前記識別子レジスタに書き込むと共に前記オフセットカウンタを初期値にリセットし、読み出された仮想マシン命令が前記非分岐命令である場合には、前記オフセットカウンタを増加させ、前記読み出し手段は、前記解読実行手段によって更新されたプログラムカウンタの識別子とオフセットに基づいて次に実行すべき仮想マシン命令を読み出すこととすることもできる。

## 【0247】

これによって、各命令ブロックは識別子レジスタの値のみによって特定され、各仮想マシン命令の相対位置はオフセットカウンタのみによって特定されるので、従来のセグメント方式によるアドレス変換技術を利用することができる。

また、前記解読実行手段はさらに、対象とする全ての種類の仮想マシン命令それぞれに対応する実マシン命令列を記憶する実マシン命令列記憶部を備え、前記命令記憶手段の各命令ブロックはさらに、その命令ブロックの非分岐命令領域及び分岐命令領域に格納された仮想マシン命令それぞれに対応する実マシン命令列を前記実マシン命令列記憶部に記憶された実マシン命令列から特定するためのデコードデータを含むデコードデータ列を格納したデコードデータ列領域を含み、前記読み出し手段は、デコードデータ列が格納された命令ブロックについては前記仮想マシン命令に代えて前記デコードデータを読み出し、デコードデータ列が格納されていない命令ブロックについては仮想マシン命令を読み出した後に、その仮想マシン命令に対応する実マシン命令列を前記実マシン命令列記憶部に記憶された実マシン命令列から特定するためのデコードデータを生成し、前記解読実行手段は、前記読み出し手段から読み出し又は生成されたデコードデータによって特定される実マシン命令列を前記実マシン命令列記憶部から読み出して実行することとすることもできる。

## 【0248】

これによって、仮想マシンプログラムを命令ブロック単位に分割して得られる上記効果に加えて、デコードデータ列が格納された命令ブロックについては、仮想マシン命令に代えてデコードデータが直接に実行されるので、解読時間が不要となり、仮想マシンの実行速度は向上されるという効果がある。

また、前記命令記憶手段の各命令ブロックはさらに、その命令ブロックのデコードデータ列領域に前記デコードデータが格納されているか否かを示すフラグを格納したフラグ領域を含み、前記解読実行手段はさらに、分岐命令を実行した場合には、その分岐先となる命令ブロックのフラグ領域に格納されたフラグを読み出して保持するカレントフラグ記憶部を備え、前記読み出し手段は、前記カレントフラグ記憶部に保持されたフラグに基づいて、デコードデータの読み出し又は仮想マシン命令の読み出しを行うとすることもできる。

## 【0249】

これによって、命令ブロックごとにデコードデータ列が格納されているか否かのフラグは、その命令ブロックから読み出されて仮想マシンに保持されるので、その命令ブロックの実行においては、仮想マシン命令ごとに命令ブロック中のフラグを参照する必要はなくなる。

また、前記命令記憶手段の各命令ブロックはさらに、その命令ブロックのデコードデータ列領域に前記デコードデータが格納されているか否かを示すフラグを格納したフラグ領域を含み、前記解読実行手段はさらに、分岐命令を実行した場合にその分岐先となる命令ブロックのフラグ領域に格納されたフラグを参照することにより、その命令ブロックにはデコードデータが格納されていないと判定したときには、その命令ブロックに格納されている仮想マシン命令列を読み出して対応するデコードデータ列に変換し、その命令ブロックのデコードデータ列領域に書き込むデコードデータ列書き込み部を備えとすることもできる。

## 【0250】

これによって、デコードデータ列を格納していない命令ブロックであっても、初めての実行時にデコードデータ列が生成されて格納されるので、ループ処理等のように繰り返して実行される場合には、2回目以降の実行時における速度が向上される。

また、本発明に係る仮想マシンは、実マシンによる制御の下で仮想マシン命令を実行する仮想マシンであって、実行対象となる仮想マシン命令列を圧縮符号で記憶する命令記憶手段と、前記命令記憶手段から次に実行すべき仮想マシン命令の圧縮符号を読み出し、対応する仮想マシン命令に復号する読み出し手段と、復

号された仮想マシン命令に対応する演算処理を特定し実行する解読実行手段とを備え、前記命令記憶手段は、前記仮想マシン命令列を構成する基本ブロックそれぞれに対応する命令ブロックの集まりからなり、各命令ブロックは、その命令ブロックの前記命令記憶手段における先頭位置を特定するための識別子を格納した識別子領域と、対応する基本ブロックの非分岐命令だけを格納した非分岐命令領域と、対応する基本ブロックの分岐命令を格納した分岐命令領域とを含み、前記分岐命令領域に格納された分岐命令は、その分岐先が前記識別子によって指定され、前記解読実行手段は、復号された仮想マシン命令が前記分岐命令である場合には、その分岐先として指定されている識別子に対応する命令ブロックの非分岐命令領域の先頭に実行制御を分岐させることを特徴とする。

## 【0251】

これによって、実行対象となる仮想マシンプログラムは、基本ブロックを単位とし、かつ、圧縮符号化されて命令記憶手段に格納され、解読実行手段によって復号化され実行されるので、圧縮符号の途中に分岐することに伴う不具合を生じることのない仮想マシンが実現される。

また、前記命令記憶手段の各命令ブロックにはさらに、その命令ブロックに格納された仮想マシン命令の圧縮符号を伸長復号するための情報テーブルであって、対応する圧縮符号と仮想マシン命令との組の集まりからなる復号テーブルを格納した復号テーブル領域を含み、前記読み出し手段は、次に実行すべき仮想マシン命令が属する命令ブロックの復号テーブルを参照しながら、前記命令記憶手段から読み出した圧縮符号を対応する仮想マシン命令に復号するとすることもできる。

## 【0252】

これによって、命令ブロックごとに復号テーブルが格納され参照されるので、命令ブロックごとに異なる種類の圧縮符号化を施した場合であっても、適正に復号されて実行されることが保証される。

また、本発明に係るその場式コンパイラは、実マシンによる制御の下で仮想マシン命令を実行する仮想マシンと共に用いられ、実行対象となる一部の仮想マシン命令列をその実行に先立って実マシン命令列に変換するその場式コンパイラで

あって、前記仮想マシン命令列を構成する仮想マシン命令それぞれについて、前記仮想マシン命令列を基本ブロックに分割した場合の各基本ブロックの先頭となる仮想マシン命令であるか否かを示すブロック先頭情報の入力を受け付けるブロック先頭情報獲得手段と、前記仮想マシン命令列を構成する仮想マシン命令それぞれについて、対応する実マシン命令列に変換する変換手段と、前記変換手段によって得られた実マシン命令列における実マシン命令であって、前記ブロック先頭情報獲得手段が受け付けたブロック先頭情報によって特定された基本ブロックの先頭となる仮想マシン命令に対応する実マシン命令が、前記実マシンによってアドレッシングされ得ない位置に配置される命令であるか否かを判定する分岐違反判定手段と、前記分岐違反判定手段によって前記実マシン命令が前記位置に配置される命令であると判定された場合に、前記実マシン命令が前記位置に配置されないよう 1 個以上の無動作命令を前記実マシン命令列に追加挿入して出力する出力手段とを備えることを特徴とする。

#### 【0253】

これによって、従来において必要とされた分岐命令の分岐先についての解析という複雑な処理を行うことなく、ジャンプ先についての境界違反を起こすことがない実マシンプログラムを高速に生成するその場式コンパイラが実現される。

ここで、前記出力手段は、前記実マシン命令が前記実マシンによってアドレッシングされ得る後続位置にずらせるために必要な個数の無動作命令を前記実マシン命令列における基本ブロックの先頭位置に追加挿入するとすることもできる。

#### 【0254】

これによって、生成される実マシンプログラムの実質的な内容を変化させることなく、ジャンプ命令に伴う境界違反の発生を回避したその場式コンパイラが実現される。

以上のように、本発明によって、仮想マシンの実行速度は高速化され、特に、異機種のコピュータ同士が接続されるネットワーク環境における共有資源の迅速かつ円滑な利用を促進させる基盤技術として、その実用的価値は極めて大きい。

#### 【図面の簡単な説明】

【図 1】

実施形態 1～9 に係る仮想マシンシステムが動作するコンピュータのハードウェア構成図である。

【図 2】

実施形態 1 に係る仮想マシン 100 の構成を示す機能ブロック図である。

【図 3】

図 3 (a) 及び (b) は、それぞれ先行命令情報格納部 101 及び命令格納部 102 に格納されている先行命令情報及び対応する仮想マシンコードの例を示す図である。

【図 4】

デコードテーブル 108 の内容を示す図である。

【図 5】

図 5 (a) 及び (b) は、それぞれ、上向き仮想マシン命令 "Push" 及び下向き仮想マシン命令 "Push" に対応するマイクロプログラムのリストを示す。

【図 6】

図 6 (a) 及び (b) は、それぞれ、上向き仮想マシン命令 "Add" 及び下向き仮想マシン命令 "Add" に対応するマイクロプログラムのリストを示す。

【図 7】

図 7 (a) 及び (b) は、それぞれ、上向き仮想マシン命令 "Mult" 及び下向き仮想マシン命令 "Mult" に対応するマイクロプログラムのリストを示す。

【図 8】

図 8 (a) 及び (b) それぞれは、図 6 (a) 及び図 7 (a) に示された上向き仮想マシン命令のマイクロプログラムの後半に相当するマイクロプログラム、図 6 (b) 及び図 7 (b) に示された下向き仮想マシン命令のマイクロプログラムの後半に相当するマイクロプログラムのリストを示す。

【図 9】

本仮想マシン 100 によって実行される仮想マシン命令が属する種別の変化を示す状態遷移図である。

【図 10】

デコード部 103 の動作を示すフローチャートである。

【図 11】

図 10 に示されたテーブル検索 (ステップ 4907) の詳細を示すフローチャートの前半部である。

【図 12】

図 10 に示されたテーブル検索 (ステップ 4907) の詳細を示すフローチャートの後半部である。

【図 13】

デコード部 103 が実行部 110 に順次出力するデコードデータ列の例を示す図である。

【図 14】

図 14 (a) 及び (b) は、図 13 に示されたデコードデータ列に従って実行部 110 が動作した場合における本仮想マシン 100 の内部状態の変化を示す図である。

【図 15】

実マシン 201 が通常マシンであってメモリ参照を行ってから 1 クロック後にその参照値を使用することができる場合におけるパイプラインの流れを示す図である。

【図 16】

実マシン 201 がスーパースカラマシンであってメモリ参照を行ってから 1 クロック後にその参照値を使用することができる場合におけるパイプラインの流れを示す図である。

【図 17】

実マシン 201 が通常マシンであってメモリ参照を行ってから 2 クロック後にその参照値を使用することができる場合におけるパイプラインの流れを示す図である。

【図 18】

実マシン 201 がスーパースカラマシンであってメモリ参照を行ってから 2 クロック後にその参照値を使用することができる場合におけるパイプラインの流れ



を示す図である。

【図 19】

実施形態 1 に係る仮想マシンコンパイラ 3400 の構成を示す機能ブロック図である。

【図 20】

命令列変換部 3402 に入力されるソースプログラム 3404 のデータ構造を示す図である。

【図 21】

図 20 における各ノードのデータ構造を示す図である。

【図 22】

命令列変換部 3402 の動作手順の概略を示すフローチャートである。

【図 23】

図 22 におけるステップ 5405 の詳細を示すフローチャートである。

【図 24】

図 23 におけるステップ 5613 の詳細を示すフローチャートである。

【図 25】

先行命令情報生成部 3401 の動作を示すフローチャートである。

【図 26】

関連付け部 3403 の動作を示すフローチャートである。

【図 27】

実施形態 2 に係る仮想マシン 3500 の構成を示す機能ブロック図である。

【図 28】

デコード部 3502 の動作のうち、テーブル検索とデコードデータの出力処理の詳細を示すフローチャートである。

【図 29】

分岐命令検出部 3505 の動作を示すフローチャートである。

【図 30】

割込み命令挿入部 3506 の動作を示すフローチャートである。

【図 31】

実施形態 3 に係る仮想マシン 3600 の構成を示す機能ブロック図である。

【図 3 2】

ブロック化部 3605 の動作を示すフローチャートである。

【図 3 3】

実施形態 4 に係る仮想マシン 3700 の構成を示す機能ブロック図である。

【図 3 4】

命令格納部 3701 のメモリマップを示す図である。

【図 3 5】

図 3 4 に示された実マシン関数テーブル 6502 の構造を示す図である。

【図 3 6】

実行部 3710 の動作を示すフローチャートである。

【図 3 7】

命令格納部 3701 のメモリマップの変形例を示す図である。

【図 3 8】

実施形態 5 に係る仮想マシン 3800 の構成を示す機能ブロック図である。

【図 3 9】

命令格納部 3801 における仮想マシンプログラムの格納状態の例を示す図である。

【図 4 0】

図 3 9 に示された仮想マシンプログラムの制御フローを示す図である。

【図 4 1】

PC 3804 によるアドレッシングのフォーマットを示す図である。

【図 4 2】

実行部 3810 における分岐先変換部 3811 の動作を示すフローチャートである。

【図 4 3】

図 3 9 に示された仮想マシンプログラムにおける論理的なアドレスと識別子とを物理的なアドレスに置き換えた図である。

【図 4 4】

実施形態 5 に係る仮想マシンコンパイラ 7660 の構成示すブロック図である。

【図 45】

分岐先変更テーブル 7663 a の構造を示す図である。

【図 46】

ブロック化部 7663 の動作を示すフローチャートである。

【図 47】

図 46 におけるステップ 7607 の詳細を示すフローチャートである。

【図 48】

図 46 におけるステップ 7704 の詳細を示すフローチャートである。

【図 49】

図 46 におけるステップ 7609 の詳細を示すフローチャートである。

【図 50】

命令ブロック単位でキャッシュさせた場合の PC 3804、命令ブロック格納領域 3852 a ~ 3852 d 及びキャッシュテーブル 8404 の関係を示す図である。

【図 51】

命令ブロック単位でキャッシュさせた場合の実行部 3810 による分岐命令の実行処理を示すフローチャートである。

【図 52】

実施形態 6 に係る仮想マシン 3900 の構成を示す機能ブロック図である。

【図 53】

命令格納部 3901 に格納されている仮想マシンプログラムの格納状態の例を示す図である。

【図 54】

デコード部 3902 の動作を示すフローチャートである。

【図 55】

実行部 3910 の動作を示すフローチャートである。

【図 56】

実行部 3910 が分岐命令を実行した場合のデコード部 3902 に対する制御を示すフローチャートである。

【図 57】

実施形態 7 に係る仮想マシン 4000 の構成を示す機能ブロック図である。

【図 58】

本仮想マシン 4000 が分岐命令を実行した場合におけるデコード命令列書込み部 4008、カレントフラグ読出制御部 3912 及び分岐先変換部 3811 における動作を示すフローチャートである。

【図 59】

図 58 におけるステップ 9110 の詳細を示すフローチャートである。

【図 60】

実行部 3910 から見たデコード部 4002 の動作を示すフローチャートである。

【図 61】

実施形態 8 に係る仮想マシン 4100 の構成を示す機能ブロック図である。

【図 62】

図 62 (a) は、展開情報格納領域 4157a～4157d に格納された復号化テーブルの例を示し、図 62 (b) は、図 62 (a) に示された復号化テーブルにおける符号の規則性を示す図である。

【図 63】

命令格納部 4101 に格納されている仮想マシンプログラムの格納状態の例を示す図である。

【図 64】

デコード部 4102 の動作を示すフローチャートである。

【図 65】

図 64 におけるステップ 9602 の詳細を示すフローチャートである。

【図 66】

実施形態 9 に係るその場式コンパイラ 4300 を含むコンパイラシステム全体の構成を示す機能ブロック図である。

【図 67】

ブロック先頭情報生成部 4321a の動作を示すフローチャートである。

【図 68】

実マシン命令変換部 4301、分岐位置補正部 4302 及び実マシンアドレス保持部 4303 の動作を示すフローチャートである。

【図 69】

ブロック先頭情報生成部 4321a によって生成されるブロック先頭情報、分岐位置補正部 4302 によって生成される実マシン命令 "Nop" のタイミング、その他の関連情報を示すテーブルである。

【図 70】

本発明に係る仮想マシンが実行する仮想マシン命令のフォーマットの変形例を示す図である。

【図 71】

従来のスタック型仮想マシン 4400 の構成を示す機能ブロック図である。

【図 72】

従来及び本発明に係る仮想マシンの命令セットを説明するための図である。

【図 73】

図 71 に示されたデコードテーブル 4406 の内容を示す図である。

【図 74】

図 71 に示されたマイクロプログラム記憶部 4411 に格納されているマイクロプログラムのリストを示す図である。

【図 75】

従来及び本発明に係る実マシン命令の意味を示す図である。

【図 76】

図 71 に示されたデコード部 4402 の動作を示すフローチャートである。

【図 77】

図 76 のフローチャートにおけるステップ 4506 の詳細を示すフローチャートである。

【図 78】

デコード部 4402 からのデコードデータがバッファを介して実行部 4410 に渡される場合におけるデコード部 4402 の動作を示すフローチャートである。

【図 79】

図 71 に示された実行部 4410 の動作を示すフローチャートである。

【図 80】

図 80 (a) は、サンプルプログラムのリストを示し、図 80 (b) は、そのプログラムが指示する演算式「 $2 \times (3+4)$ 」を示し、図 80 (c) は、デコード部 4402 から順次出力されるデコードデータを示す図である。

【図 81】

図 80 (c) に示されたデコードデータ列に従って実行部 4410 が動作した場合における従来の仮想マシンの内部状態の変化を示す図である。

【図 82】

TOS変数を用いる従来の仮想マシンにおけるマイクロプログラムのリストを示す図である。

【図 83】

図 82 (a) ~ (d) に示されたマイクロプログラムを備える従来の仮想マシンによって図 80 (a) の仮想マシンプログラムが実行された場合の仮想マシンの内部状態の変化を示す図である。

【図 84】

パイプラインの各ステージに対する省略記号を説明する図である。

【図 85】

通常マシンのパイプラインの理想的な流れを示す図である。

【図 86】

スーパースカラマシンのパイプラインの理想的な流れを示す図である。

【図 87】

通常マシンにおいて、パイプラインにハザードが生じた場合のパイプラインの流れを示す図である。

【図 88】

スーパースカラマシンにおいて、パイプラインにハザードが生じた場合のパイプラインの流れを示す図である。

【図 89】

図 87 の場合において、メモリ参照が行われてからその値が次に使用されるまでに 2 クロックの時間を必要とするときのパイプラインの流れを示す図である。

【図 90】

図 88 の場合において、メモリ参照が行われてからその値が次に使用されるまでに 2 クロックの時間を必要とするときのパイプラインの流れを示す図である。

【図 91】

通常マシンにおいて、命令 A 及び命令 A 2 がレジスタでジャンプ先を指定する命令である場合のパイプラインの流れを示す図である。

【図 92】

スーパースカラマシンにおいて、命令 A 及び命令 A 2 がレジスタでジャンプ先を指定する命令である場合のパイプラインの流れを示す図である。

【図 93】

第 1 の従来技術に係る仮想マシンが図 80 (a) に示された仮想マシンプログラムを実行した場合であって、メモリ参照が行われてからその値が次に使用されるまでに 1 クロックの時間で済む場合における通常マシンのパイプラインの流れを示す。

【図 94】

図 93 と同様の場合であって、スーパースカラマシンのパイプラインの流れを示す。

【図 95】

メモリ参照が行われてからその値が次に使用されるまでに 2 クロックの時間を必要とする場合における通常マシンのパイプラインの流れを示す。

【図 96】

図 95 と同様の場合であって、スーパースカラマシンのパイプラインの流れを示す。

【図 97】

サンプル仮想マシンプログラムのリストを示す図である。

【図 9 8】

図 9 7 に示されたサンプル仮想マシンプログラムのフローチャートである。

【図 9 9】

従来のその場式コンパイルにおいて使用される変換テーブルである。

【図 100】

図 9 9 に示された変換テーブルを用いて図 9 7 に示された仮想マシンプログラムをコンパイルした場合に得られる実マシンプログラムのコード配置の様子を図である。

【図 101】

圧縮符号化テーブルの例を示す図である。

図 101 (a) は、圧縮符号化テーブルの例を示し、図 101 (b) は、誤って復号化される場合の例を示す図である。

【図 102】

キャッシュメモリを備える従来の仮想マシンにおいて生じうる問題点を説明するための図である。

【図 103】

図 9 7 に示された仮想マシンプログラムがキャッシュメモリに置かれた場合における各キャッシュブロックを区切る境界線 A, B, C を示す図である。

【符号の説明】

100	仮想マシン
101	先行命令情報格納部
102	命令格納部
103	デコード部
104	先行命令情報読込み部
105	命令読込み部
106	検索部
107	PC
108	デコードテーブル



110	実行部
111	マイクロプログラム記憶部
112	SP
120	スタック
121	TOS変数
122	SOS変数
123	メモリスタック
200	コンピュータ
201	実マシン
202	メモリ
203	キーボード
204	マウス
205a~205c	内部バス
206	ディスプレイ
207	ハードディスク
208	ネットワークカード
3400	仮想マシンコンパイラ
3401	先行命令情報生成部
3402	命令列変換部
3403	関連付け部
3404	ソースプログラム
3405	出力プログラム
3500	仮想マシン
3502	デコード部
3505	分岐命令検出部
3506	割込み命令挿入部
3507	割込み状態格納部
3510	割込み制御部
3515	実行部

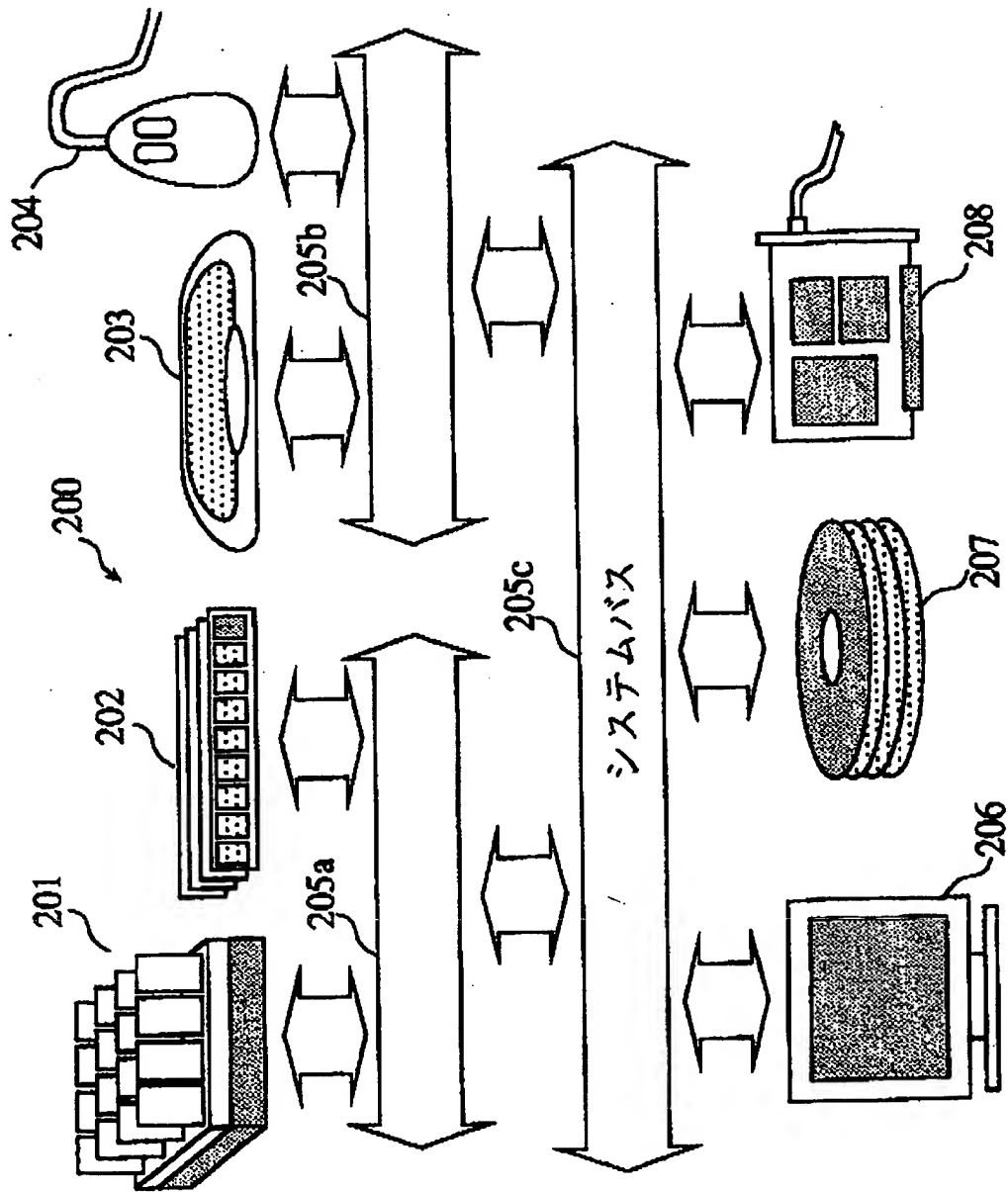
3516	割込み処理プログラム
3600	仮想マシン
3605	ブロック化部
3610	割込み制御部
3700	仮想マシン
3701	命令格納部
3704	領域判定部
3705	アドレス変換部
3706	実マシン関数記憶部
3710	実行部
3800	仮想マシン
3801	命令格納部
3802	デコード部
3804	PC
3804 a	識別子セグメントレジスタ
3804 b	オフセットカウンタ
3810	実行部
3811	分岐先変換部
3852 a~3852 d	命令ブロック格納領域
3853 a~3853 d	識別子格納領域
3854 a~3854 d	非分岐命令格納領域
3855 a~3855 d	分岐命令格納領域
3900	仮想マシン
3901	命令格納部
3902	デコード部
3903	命令読込み部
3907	カレントフラグ記憶部
3910	実行部
3912	カレントフラグ読出制御部

3952 a ~ 3952 d 命令ブロック格納領域  
 3953 a ~ 3953 d 識別子格納領域  
 3954 a ~ 3954 d 非分岐命令格納領域  
 3955 a ~ 3955 d 分岐命令格納領域  
 3956 a ~ 3956 d デコード命令列格納領域  
 4000 仮想マシン  
 4002 デコード部  
 4008 デコード命令列書込み部  
 4100 仮想マシン  
 4101 命令格納部  
 4102 デコード部  
 4103 命令読み込み部  
 4103 a 仮想マシン命令展開部  
 4152 a ~ 4152 d 命令ブロック格納領域  
 4153 a ~ 4153 d 識別子格納領域  
 4154 a ~ 4154 d 非分岐命令格納領域  
 4155 a ~ 4155 d 分岐命令格納領域  
 4156 a ~ 4156 d デコード命令列格納領域  
 4157 a ~ 4157 d 展開情報格納領域  
 4164 a ~ 4164 d 圧縮仮想マシンコード領域  
 4300 その場式コンパイラ  
 4301 実マシン命令変換部  
 4302 分岐位置補正部  
 4303 実マシンアドレス記憶部  
 4311 実マシン命令列  
 4320 仮想マシンコンパイラ  
 4321 出力部  
 4321 a ブロック先頭情報生成部  
 6501 仮想マシンプログラム領域

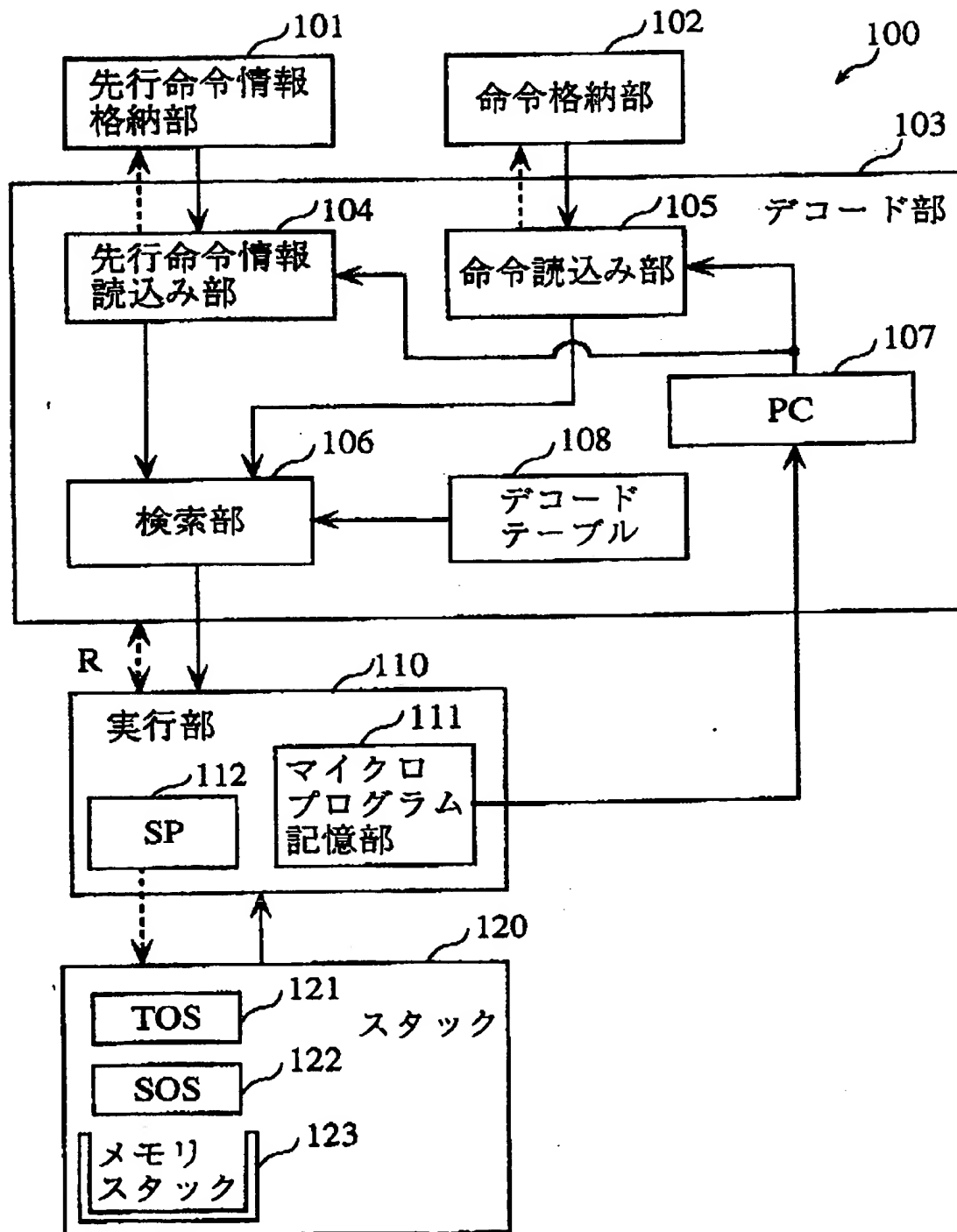
6502	実マシン関数テーブル領域
6701	メモリ属性
7650	ソースプログラム
7651	命令ブロック集合
7660	仮想マシンコンパイラ
7661	中間命令列変換部
7662	生成部
7663	ブロック化部
7663a	分岐先変更テーブル
7664	仮想マシンプログラム
8402	命令キャッシュ
8404	キャッシュテーブル
8605a~8605d	フラグ領域
8607a~8607d	実マシンコード領域

【書類名】 図面

【図 1】



【図 2】



【図 3】

101		102	
1:	U	1:	Push
2:	U	2:	2
3:	U	3:	Push
4:	U	4:	3
5:	D	5:	Push
6:	D	6:	4
7:	D	7:	Add
8:	D	8:	Mult
9:	U	9:	Pop
10:	U	10:	0
(a)		(b)	

【図 4】

仮想マシン 命令	先行命令 情報	ジャンプアドレス	オペランド 数
:	:	:	:
Push	U	<上向きPushを処理するコードへのジャンプアドレス>	1
Push	D	<下向きPushを処理するコードへのジャンプアドレス>	1
Pop	U	<上向きPopを処理するコードへのジャンプアドレス>	1
Pop	D	<下向きPopを処理するコードへのジャンプアドレス>	1
Add	U	<上向きAddを処理するコードへのジャンプアドレス>	0
Add	D	<下向きAddを処理するコードへのジャンプアドレス>	0
Sub	U	<上向きSubを処理するコードへのジャンプアドレス>	0
Sub	D	<下向きSubを処理するコードへのジャンプアドレス>	0
Inc	U	<上向きIncを処理するコードへのジャンプアドレス>	0
Inc	D	<下向きIncを処理するコードへのジャンプアドレス>	0
:	:	:	:



【図5】

(a)

仮想マシン命令(上向き) "Push"のマイクロプログラム		
1:Load	r4,r0	;TOSレジスタ(0番)の値を、SOSレジスタ ;(4番)にコピーする
2:Load	r0,[r2]	;オペランドを取り出し、 ;TOSレジスタに読み込む
3:Inc	r2	;仮想マシンのPCを一つ増やし、 ;次の命令を読み込めるようにする
4:Inc	r3	;仮想マシンのSPの値を増やす
5:Store	[r3],r4	;SOSレジスタの値を、スタックに格納する
6:Load	r1,[r2]	;仮想マシンのPCの位置の仮想 ;マシン命令(ジャンプアドレス)を ;レジスタ1番に読み込む
7:Inc	r2	;仮想マシンのPCの値を1増やす
8:Jmp	r1	;レジスタ1番で示される位置に ;無条件ジャンプする

(b)

仮想マシン命令(下向き) "Push"のマイクロプログラム		
1:Load	r4,r0	;TOSレジスタの値を、SOSレジスタに ;コピーする
2:Load	r0,[r2]	;オペランドを取り出し、 ;TOSレジスタに読み込む
3:Inc	r2	;仮想マシンのPCを一つ増やし、 ;次の命令を読み込めるようにする
6:Load	r1,[r2]	;仮想マシンのPCの位置の仮想 ;マシン命令(ジャンプアドレス)を ;レジスタ1番に読み込む
7:Inc	r2	;仮想マシンのPCの値を1増やす
8:Jmp	r1	;レジスタ1番で示される位置に ;無条件ジャンプする

【図6】

(a)	仮想マシン命令(上向き) "Add"のマイクロプログラム	
	1:Add	r0,r0,r4 ;TOSレジスタと、SOSレジスタを加え、 ;結果をTOSレジスタに代入する
	<上向きの仮想マシンジャンプコード>	

(b)	仮想マシン命令(下向き) "Add"のマイクロプログラム	
	1:Add	r0,r0,r4 ;TOSレジスタと、SOSレジスタを加え、 ;結果をTOSレジスタに代入する
	<下向きの仮想マシンジャンプコード>	

【図7】

(a)	仮想マシン命令(上向き) "Mult"のマイクロプログラム	
	1:Mult	r0,r0,r4 ;TOSレジスタと、SOSレジスタを掛け、 ;結果をTOSレジスタに代入する
	<上向きの仮想マシンジャンプコード>	

(b)	仮想マシン命令(下向き) "Mult"のマイクロプログラム	
	1:Mult	r0,r0,r4 ;TOSレジスタと、SOSレジスタを掛け、 ;結果をTOSレジスタに代入する
	<下向きの仮想マシンジャンプコード>	

【図8】

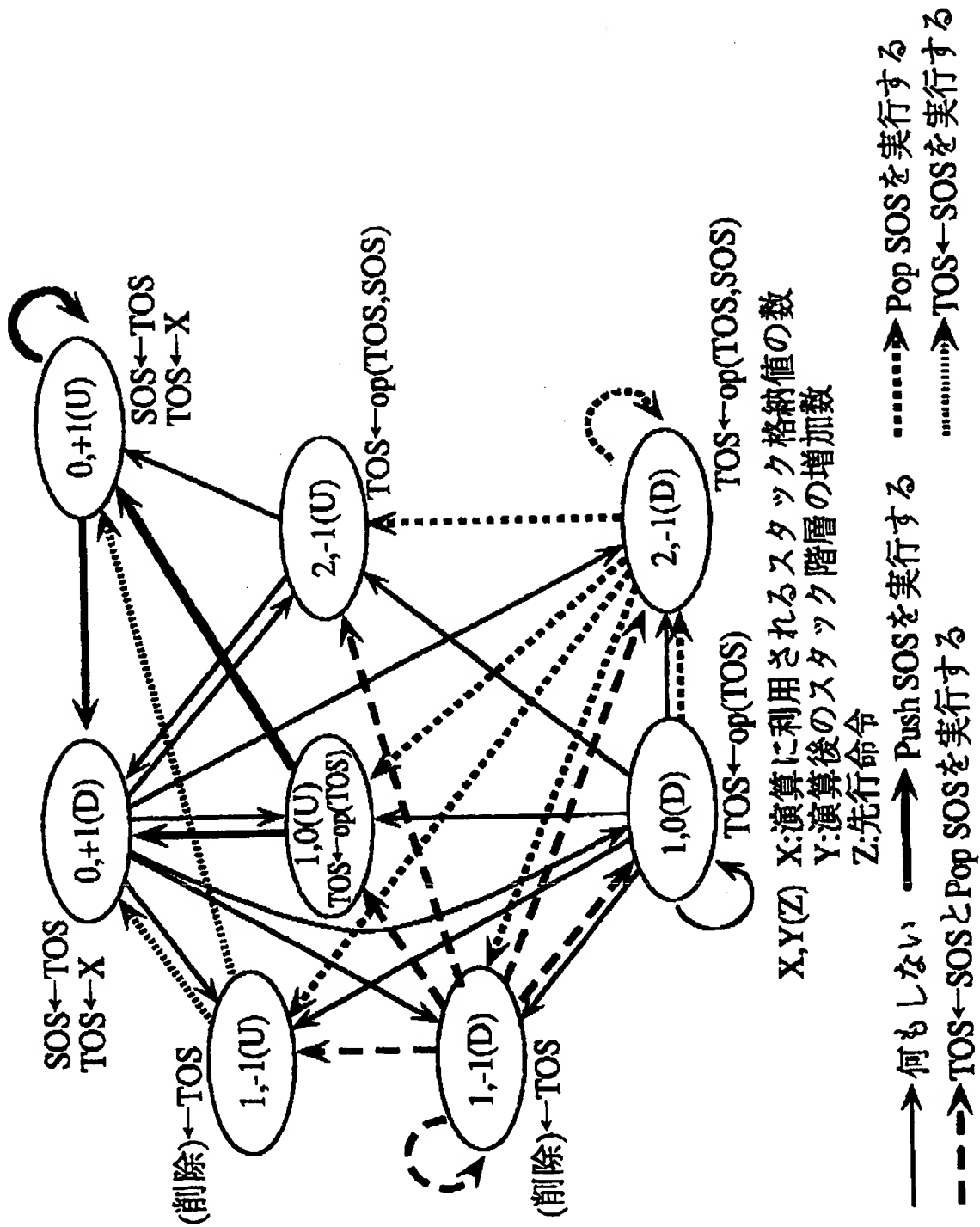
(a)

上向きの仮想マシンジャンプコード		
1:Load	r1,[r2]	;仮想マシンのPCの位置の仮想 ;マシン命令(ジャンプアドレス)を ;レジスタ1番に読み込む
2:Inc	r2	;仮想マシンのPCの値を1増やす
3:Jmp	r1	;レジスタ1番で示される位置に ;無条件ジャンプする

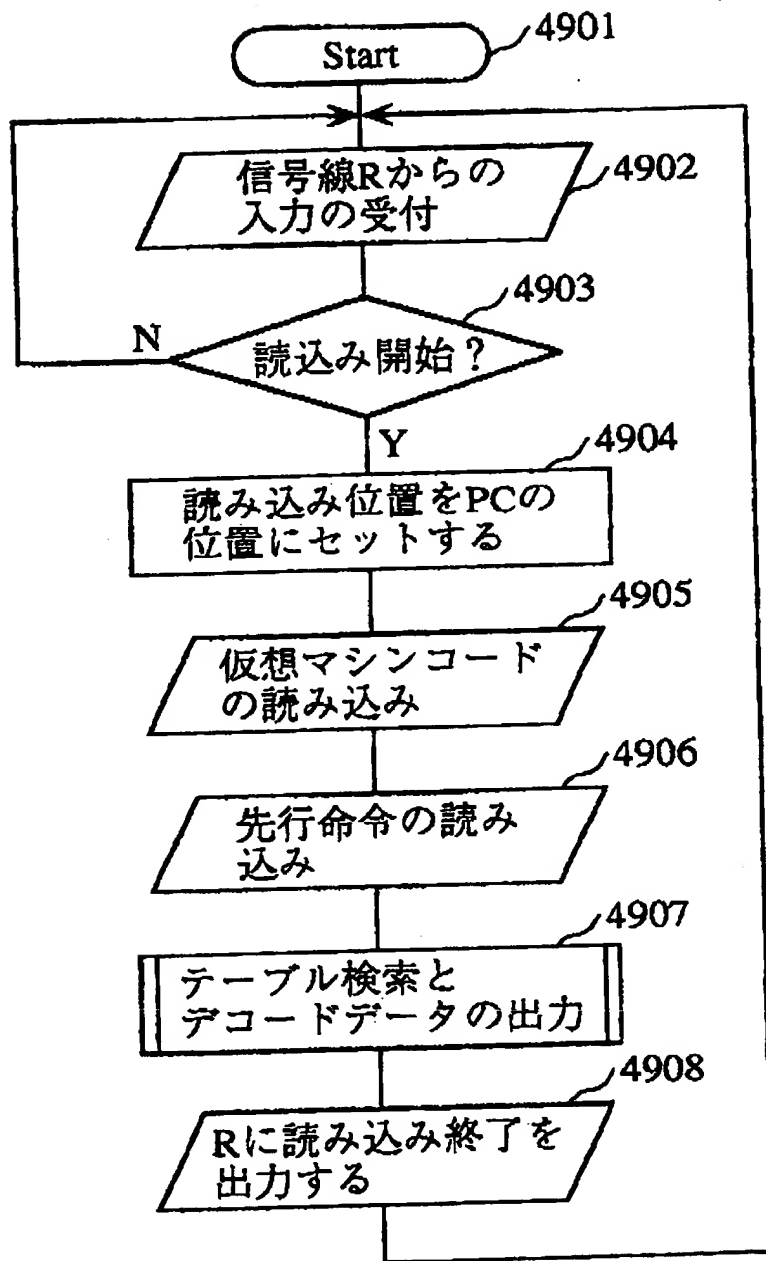
(b)

下向きの仮想マシンジャンプコード		
1:Load	r1,[r2]	;仮想マシンのPCの位置の仮想 ;マシン命令(ジャンプアドレス)を ;レジスタ1番に読み込む
2:Load	r4,[r3]	;スタックから値を取り出し、SOSレジスタに ;コピーする
3:Inc	r2	;仮想マシンのPCの値を1増やす
4:Dec	r3	;仮想マシンのSPの値を1減らす
5:Jmp	r1	;レジスタ1番で示される位置に ;無条件ジャンプする

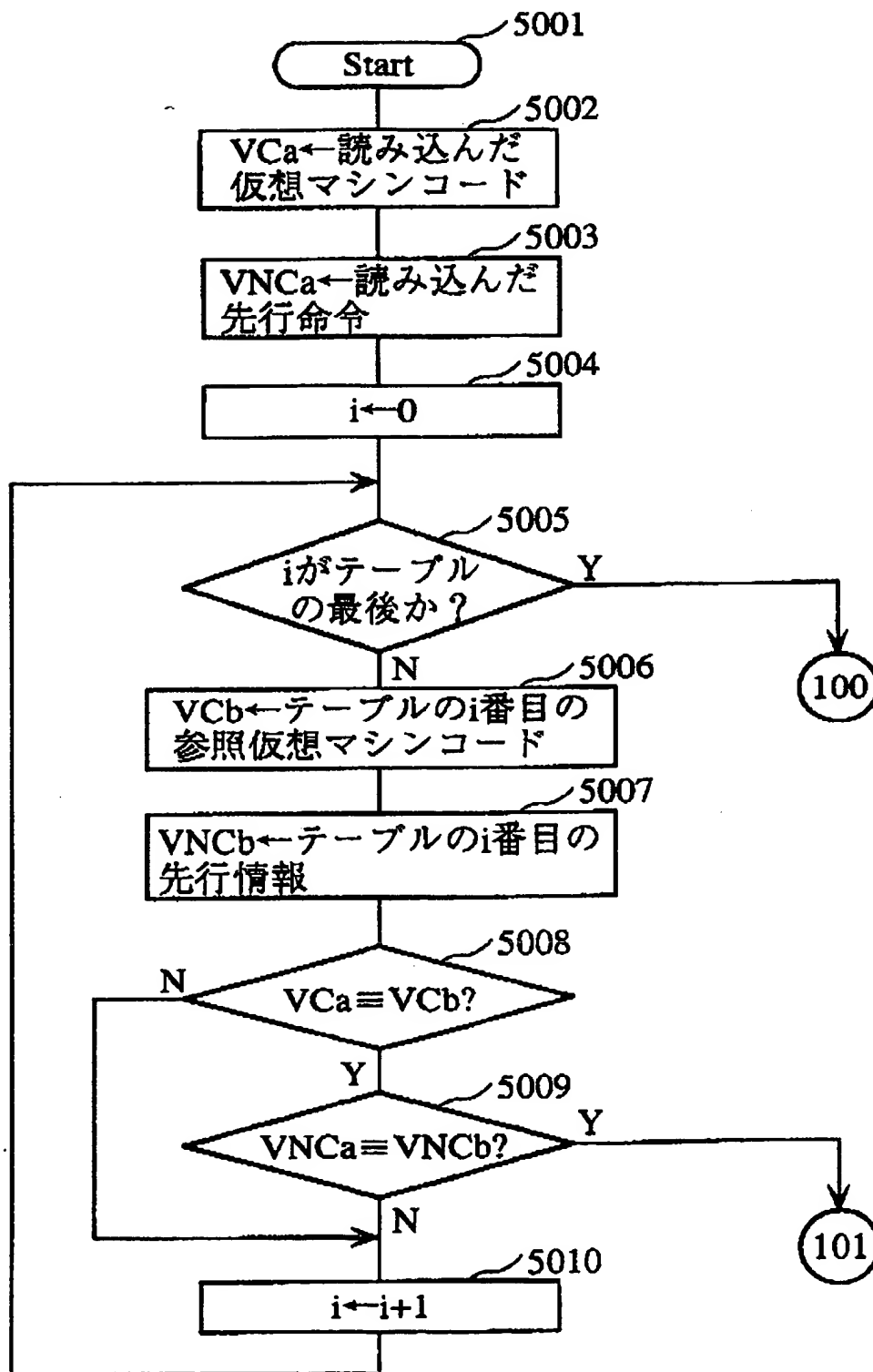
【図 9】



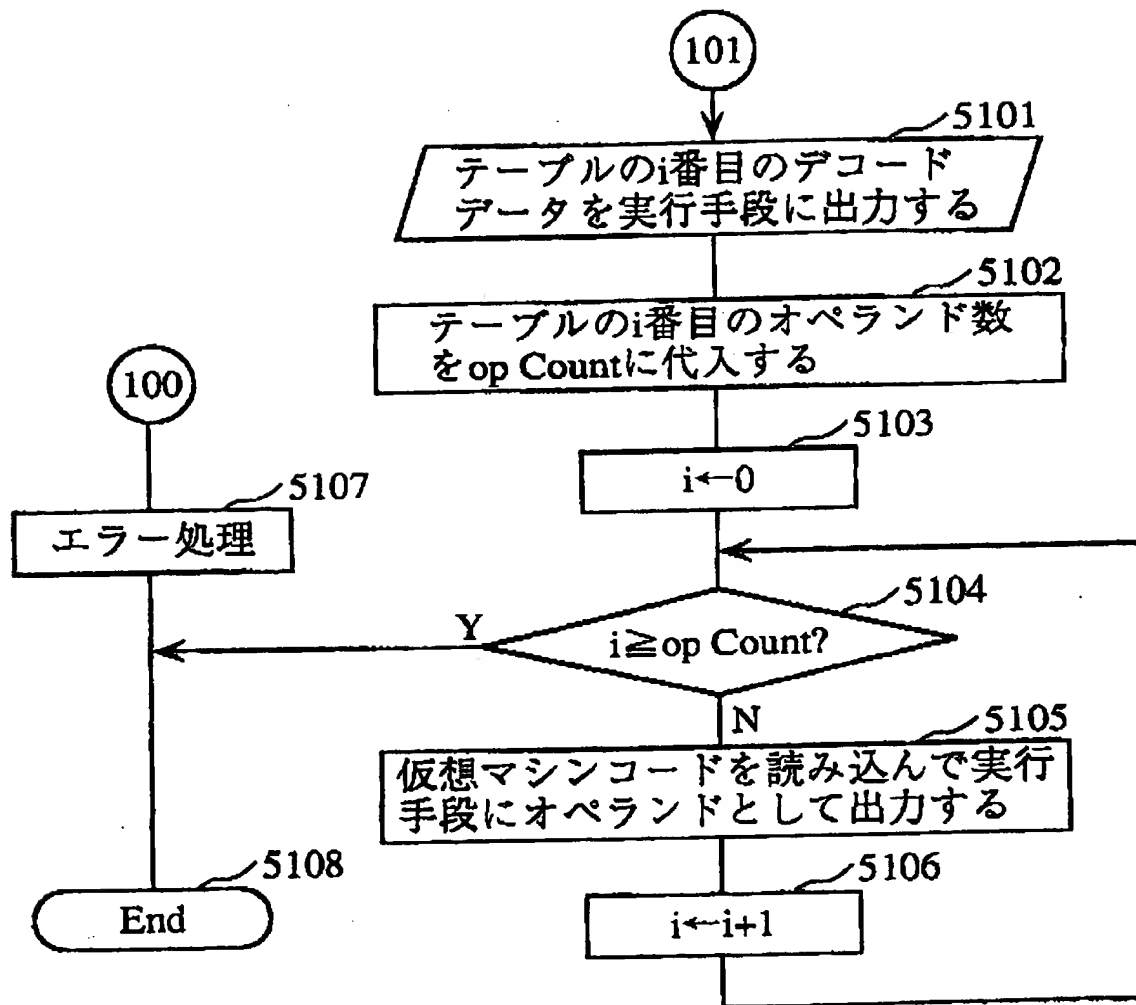
【図 10】



【図 11】



【図 12】

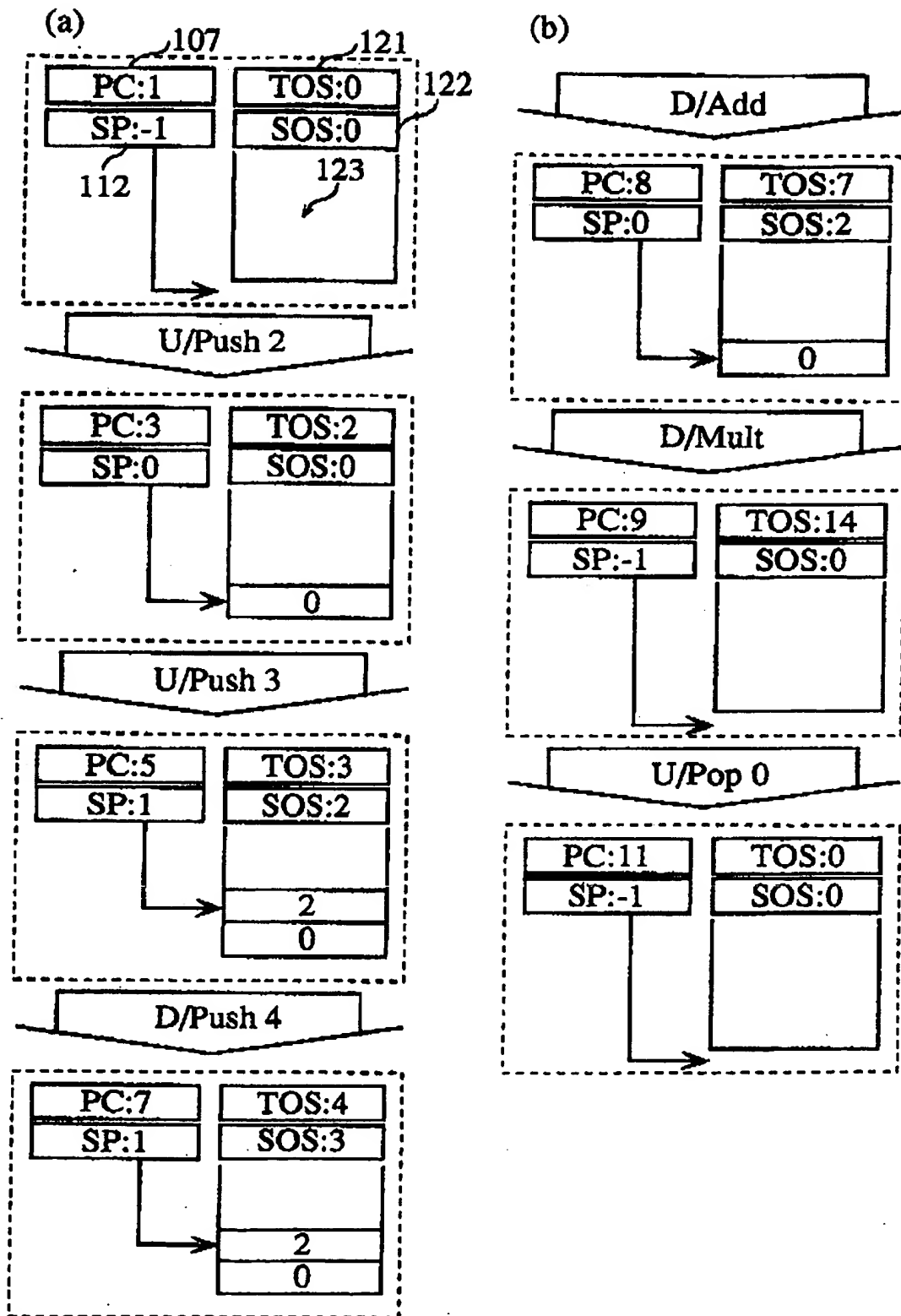


【図 13】

1:<上向きPushを処理するコードへのジャンプアドレス>
2:オペランド"2"
3:<上向きPushを処理するコードへのジャンプアドレス>
4:オペランド"3"
5:<下向きPushを処理するコードへのジャンプアドレス>
6:オペランド"4"
7:<下向きAddを処理するコードへのジャンプアドレス>
8:<下向きMultを処理するコードへのジャンプアドレス>
9:<上向きPopを処理するコードへのジャンプアドレス>
10:オペランド"0"



【図 14】



【図 15】

クロック	1	2	3	4	5	6	7	8	9	10	11
Load r1,[r2]	IF	RF	ALU	MEM	WB						
Load r4,[r3]		IF	RF	ALU	MEM	WB					
Inc r2			IF	RF	ALU	MEM	WB				
Dec r3				IF	RF	ALU	MEM	WB			
Jmp r1					IF	RF	ALU	MEM	WB		
						IF	x				
Mult r0,r0,r4							IF	RF	ALU	MEM	WB

【図 16】

クロック	1	2	3	4	5	6	7	8	9
Load r1,[r2]	IF	RF	ALU	MEM	WB				
Load r4,[r3]	IF	RF	ALU	MEM	WB				
Inc r2		IF	RF	ALU	MEM	WB			
Dec r3		IF	RF	ALU	MEM	WB			
Jmp r1			IF	RF	ALU	MEM	WB		
			IF	RF	x				
				IF	x				
				IF	x				
Mult r0,r0,r4					IF	RF	ALU	MEM	WB

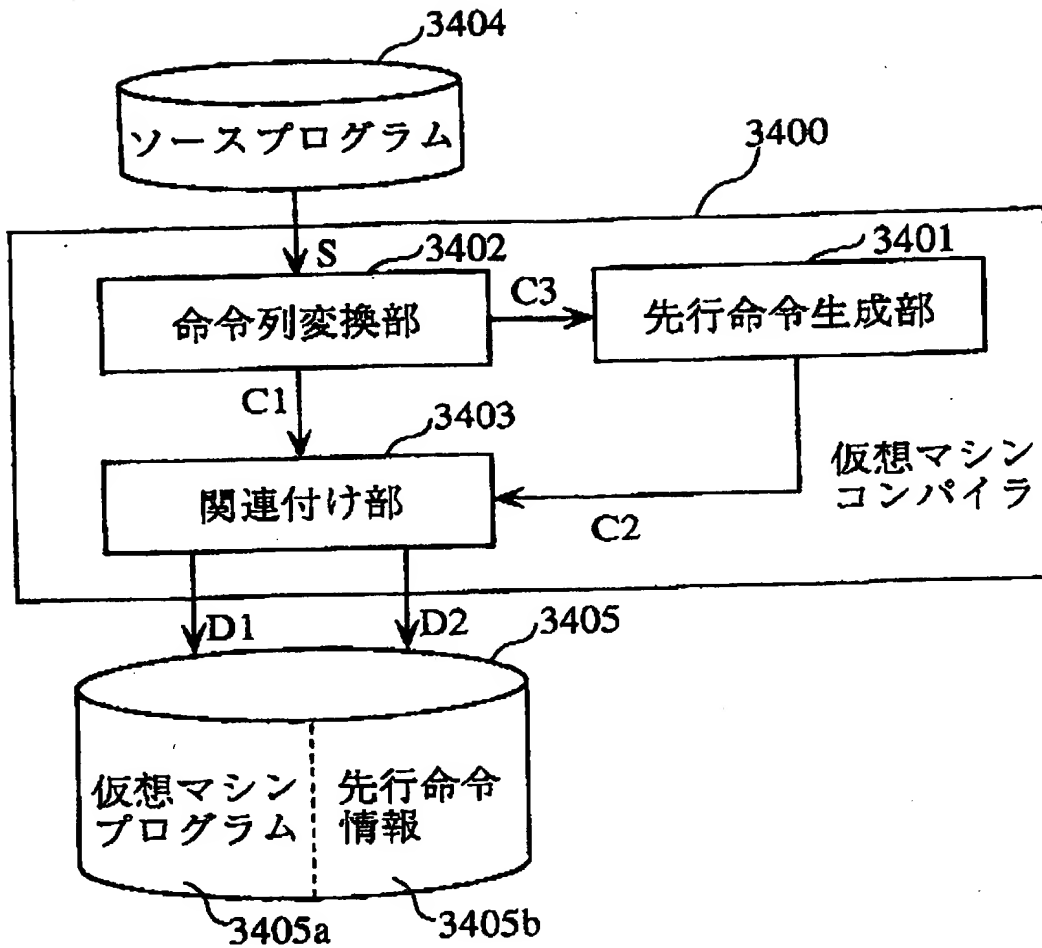
【図 17】

クロック	1	2	3	4	5	6	7	8	9	10	11
Load r1,[r2]	IF	RF	ALU	MEM	WB						
Load r4,[r3]		IF	RF	ALU	MEM	WB					
Inc r2			IF	RF	ALU	MEM	WB				
Dec r3				IF	RF	ALU	MEM	WB			
Jmp r1					IF	RF	ALU	MEM	WB		
						IF	x				
Mult r0,r0,r4							IF	RF	ALU	MEM	WB

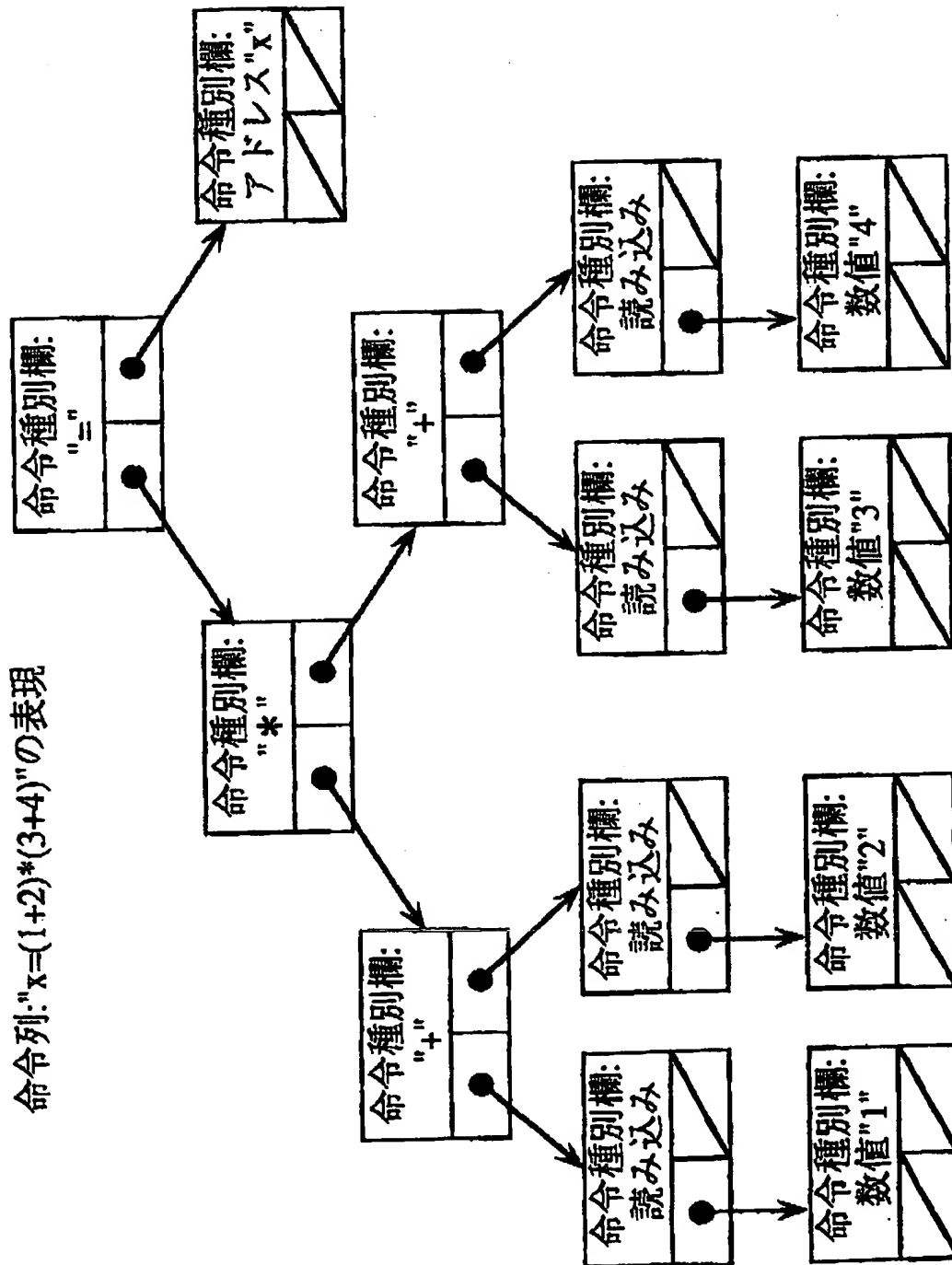
【図 18】

クロック	1	2	3	4	5	6	7	8	9	10
Load r1,[r2]	IF	RF	ALU	MEM	WB					
Load r4,[r3]	IF	RF	ALU	MEM	WB					
Inc r2		IF	RF	ALU	MEM	WB				
Dec r3		IF	RF	ALU	MEM	WB				
Jmp r1			IF	RF		ALU	MEM	WB		
			IF	RF	ALU	x				
				IF	RF	x				
				IF	RF	x				
					IF	x				
					IF	x				
Mult r0,r0,r4						IF	RF	ALU	MEM	WB

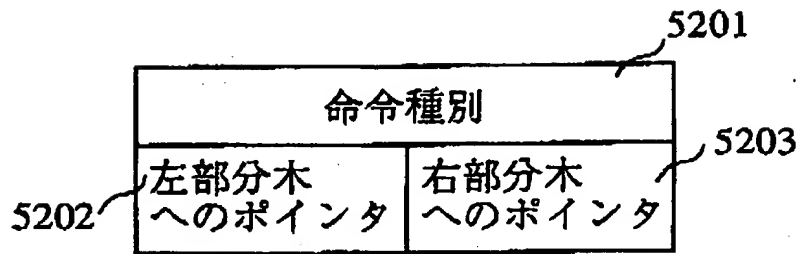
【図19】



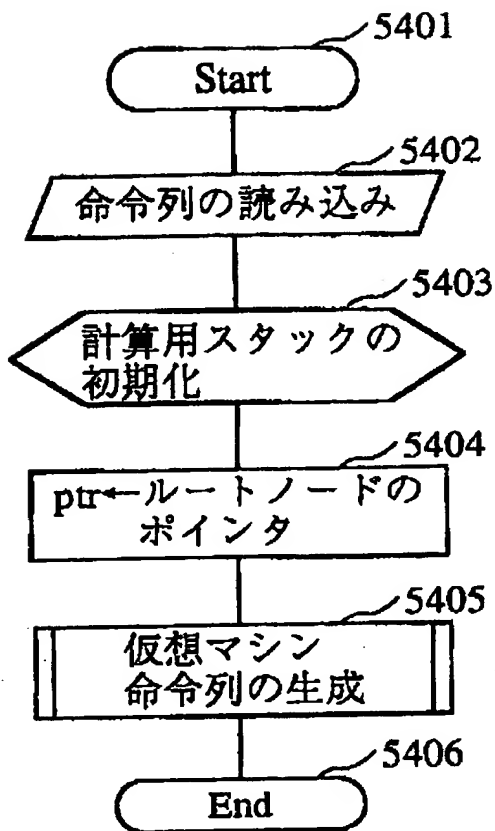
【図 20】



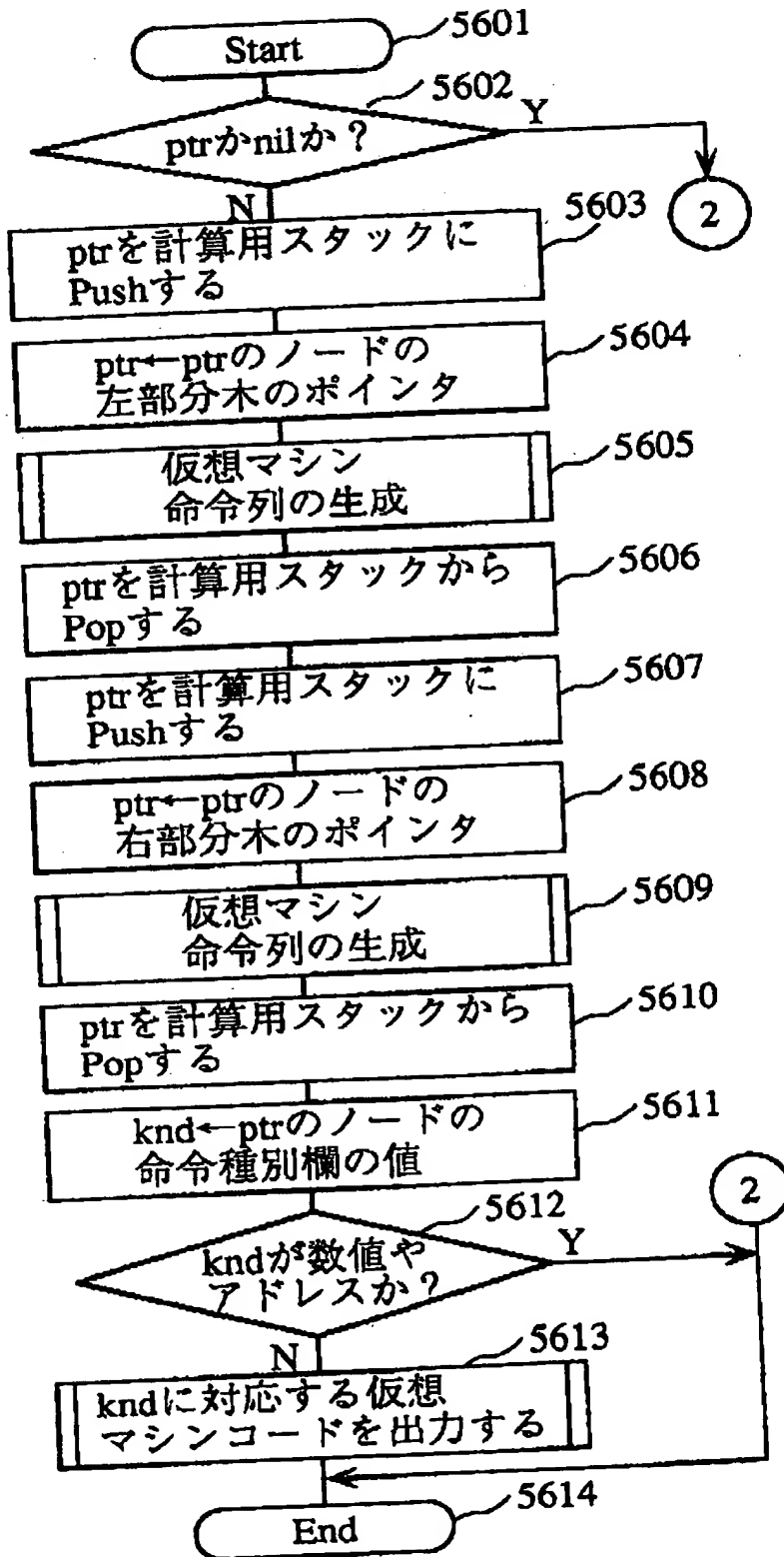
【図 21】



【図 22】

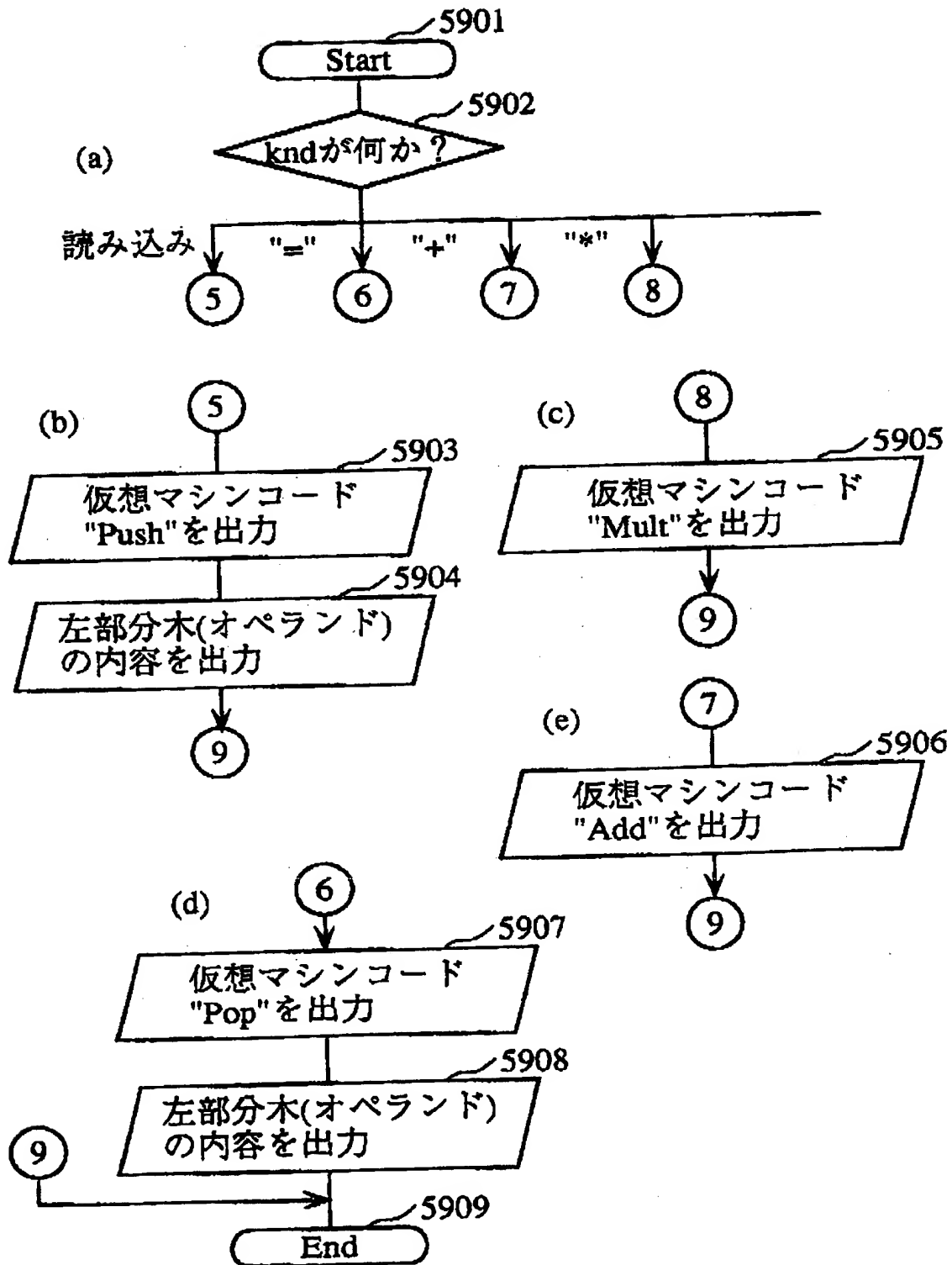


【図 23】

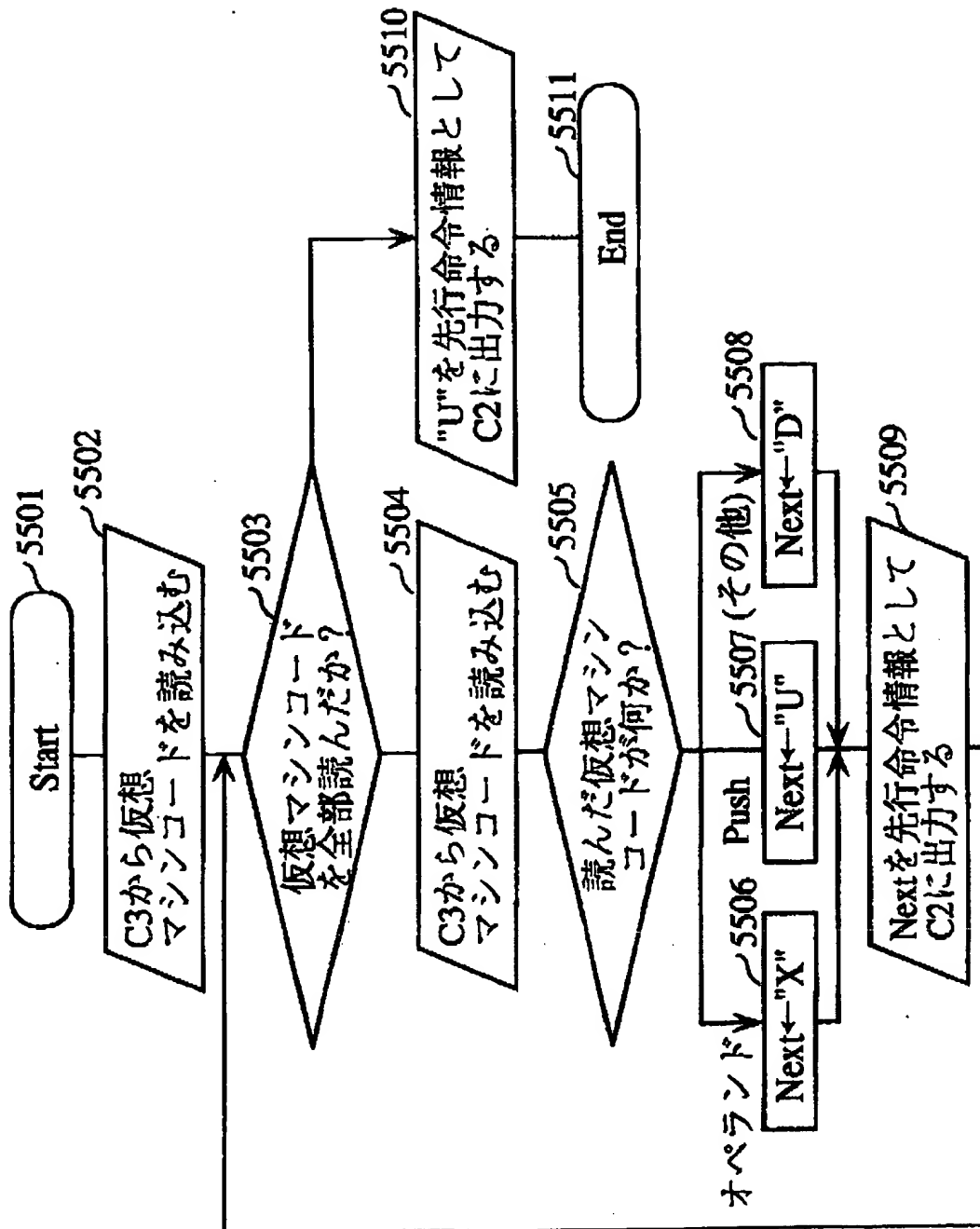




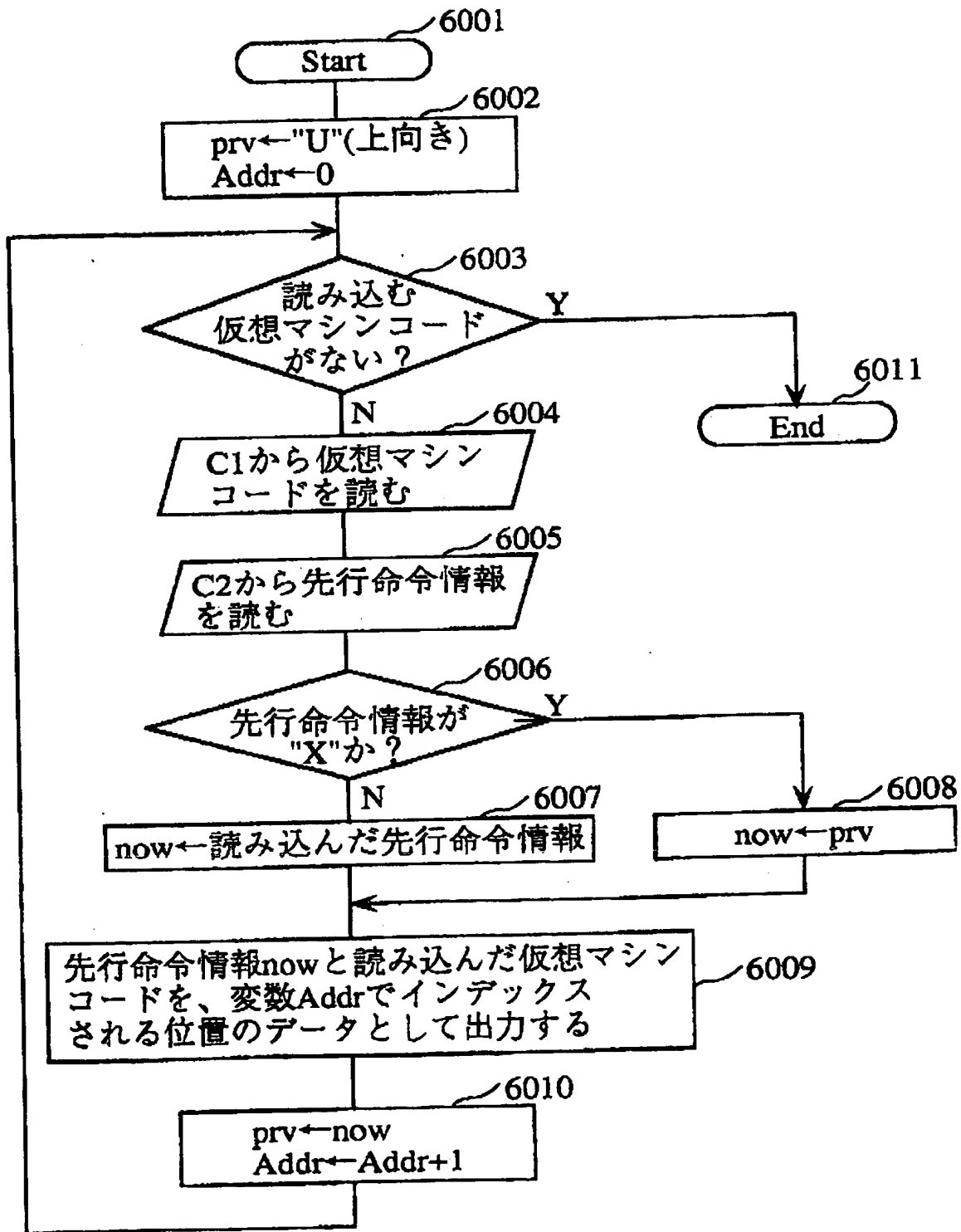
【図 24】



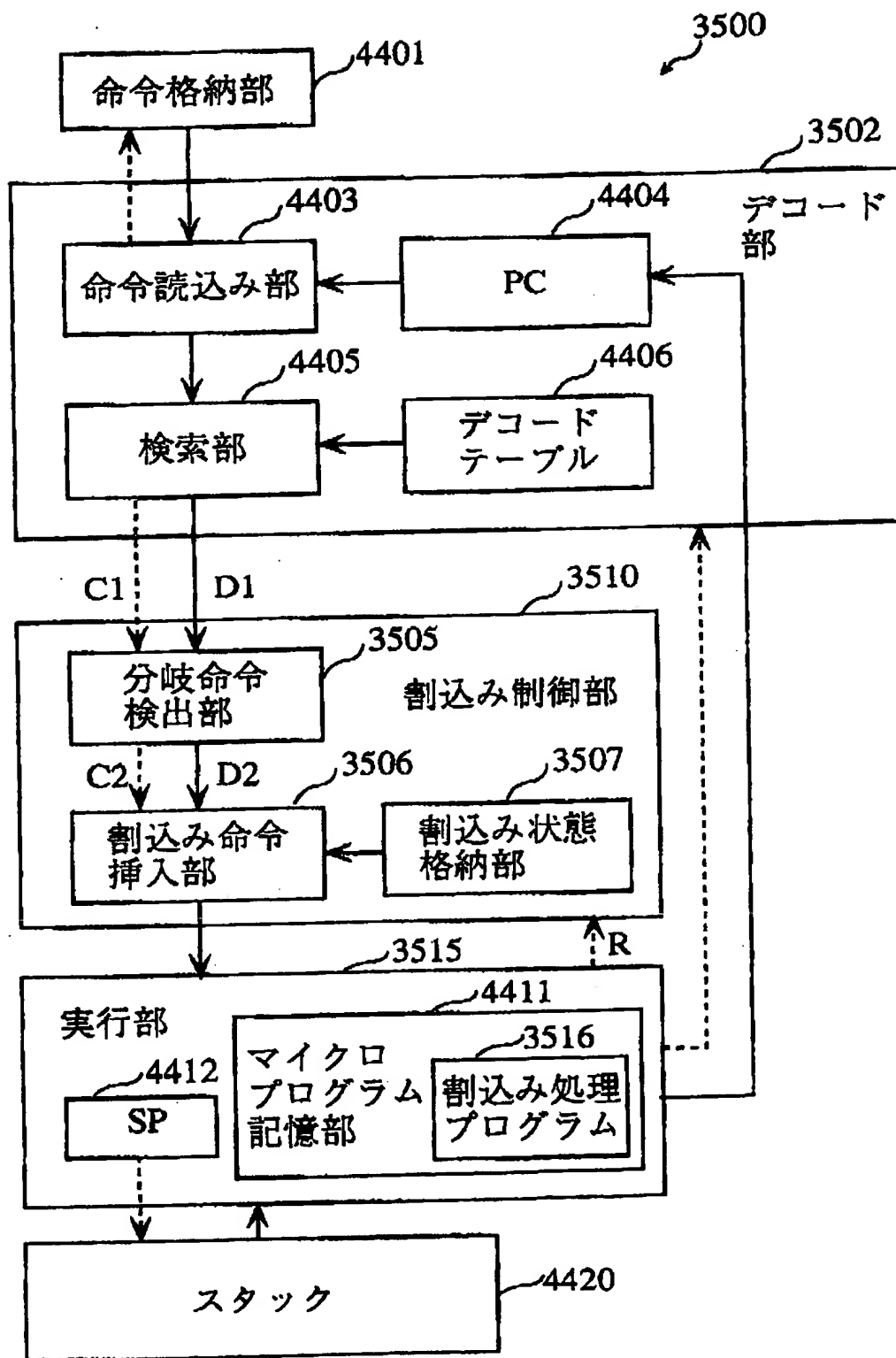
【図 2 5】



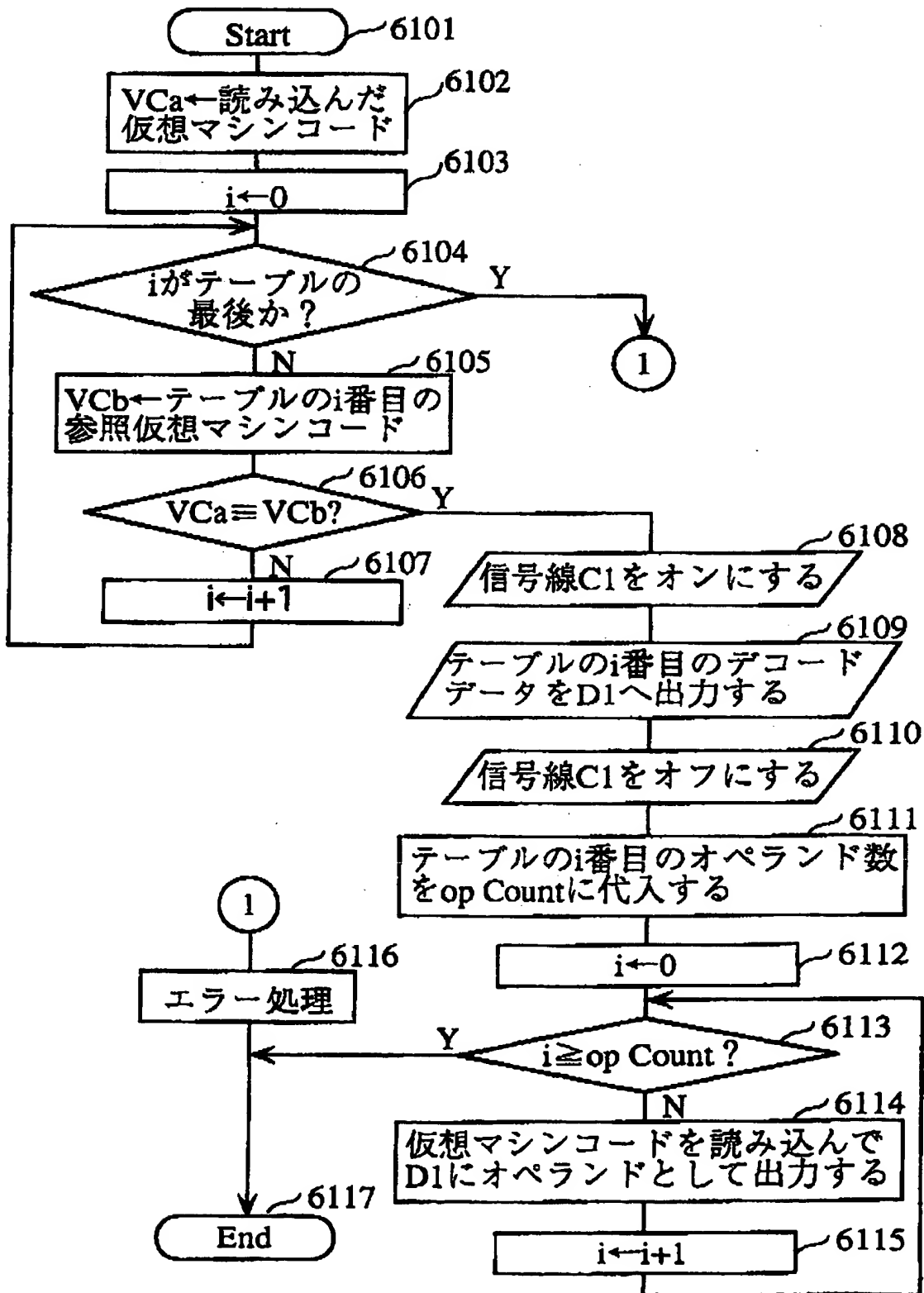
【図 26】



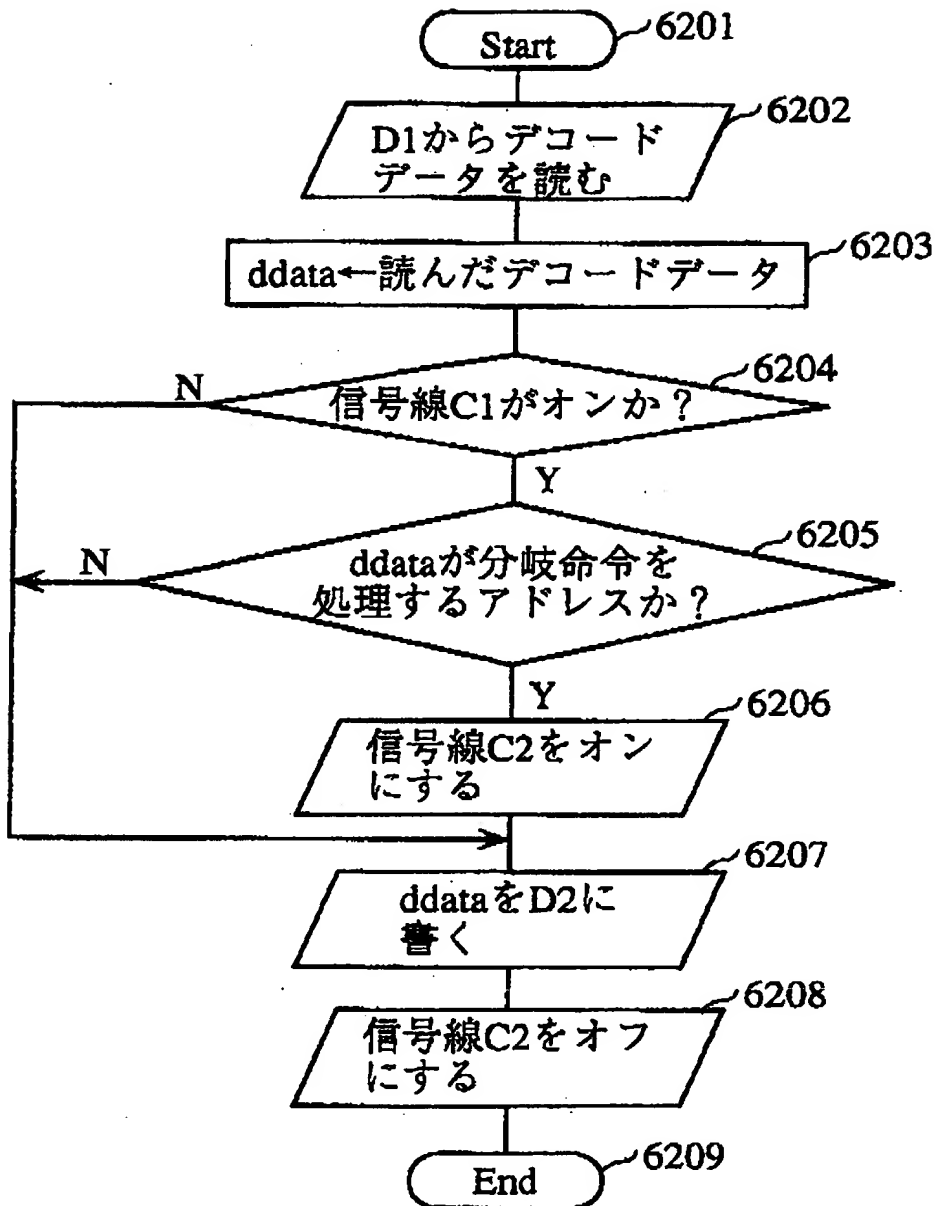
【図 27】



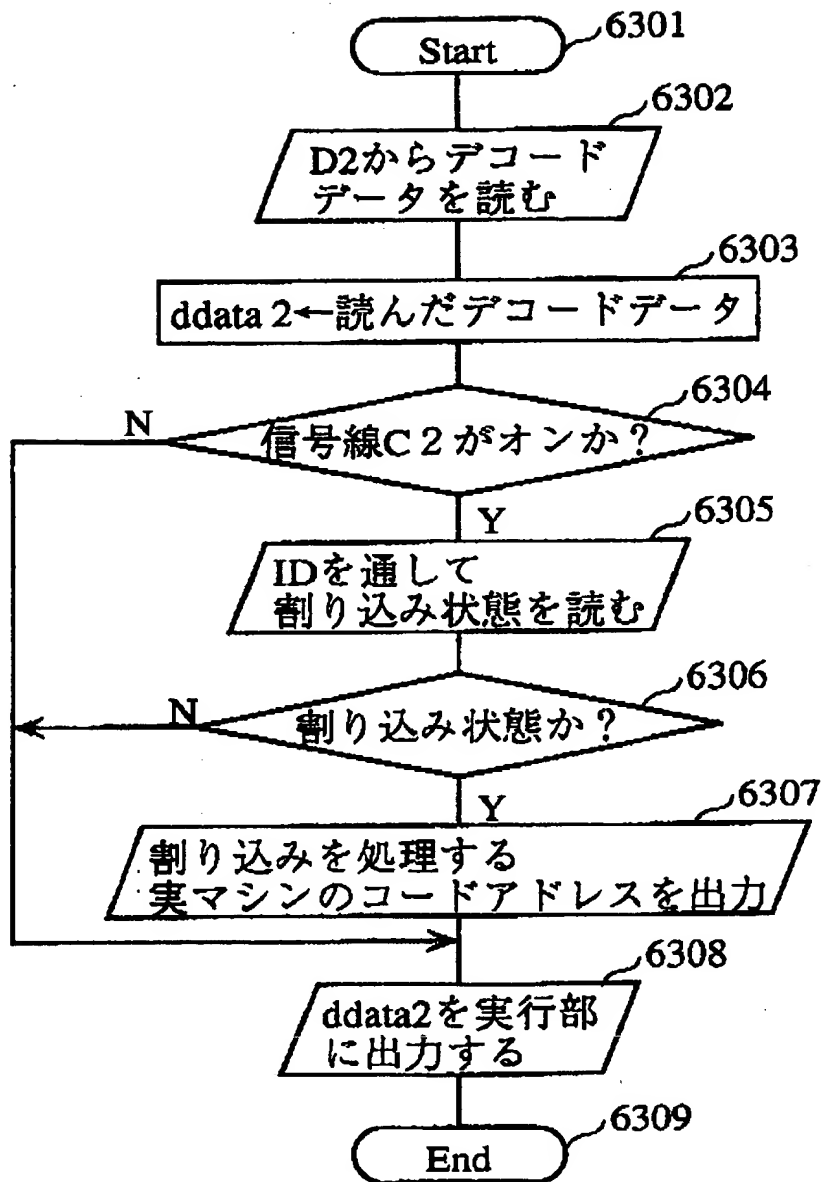
【図 28】



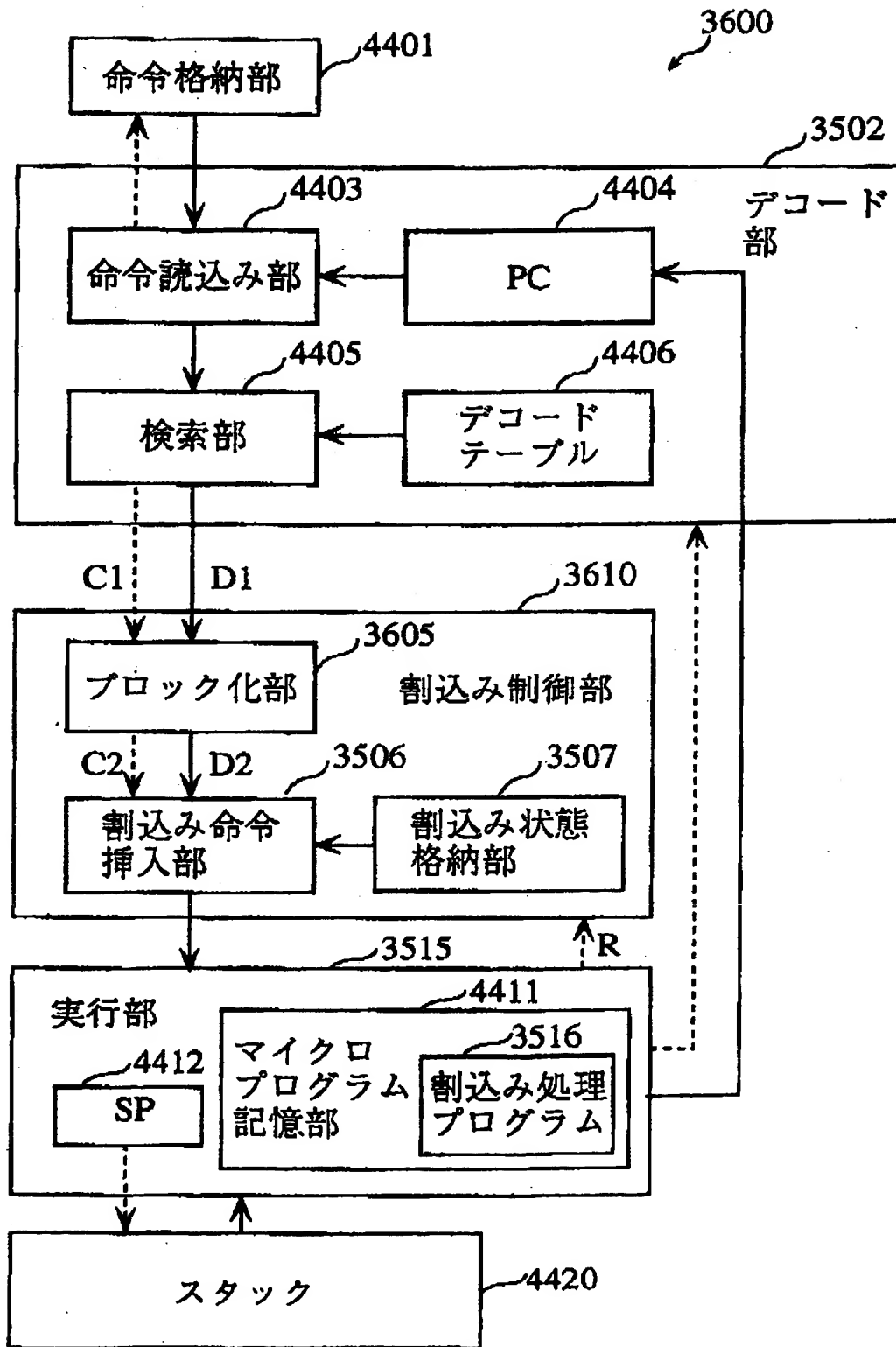
【図 29】



【図 30】

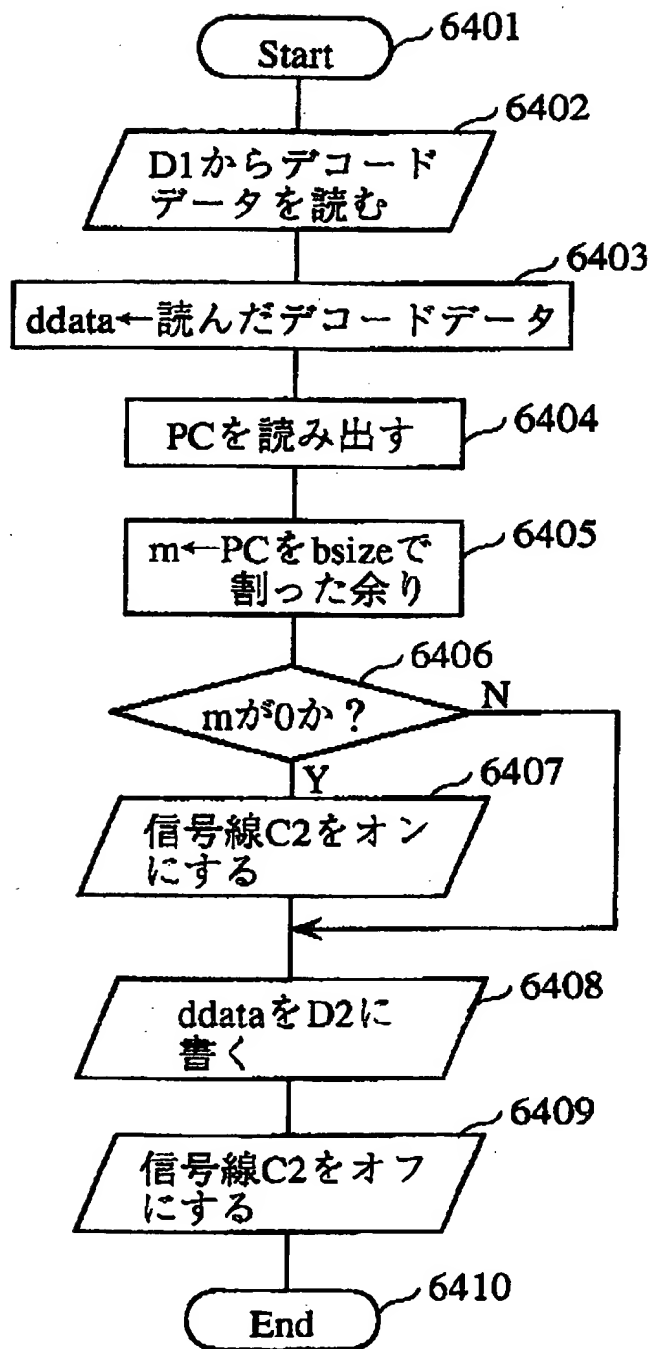


【図 31】

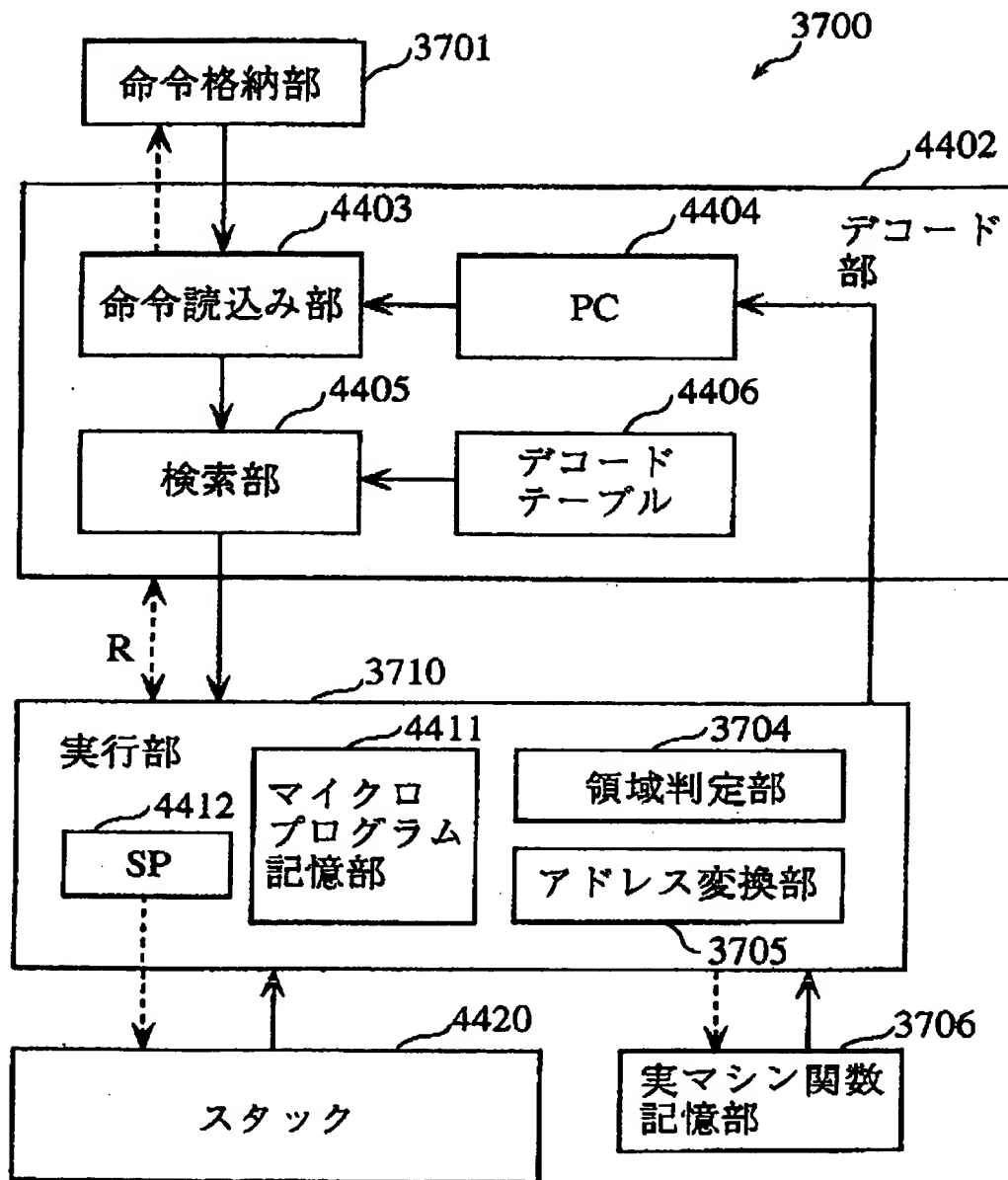




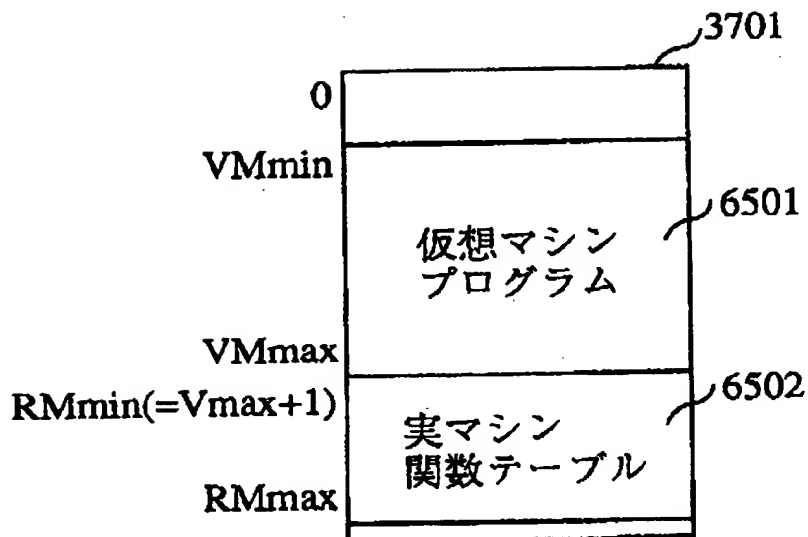
【図 32】



【図 33】



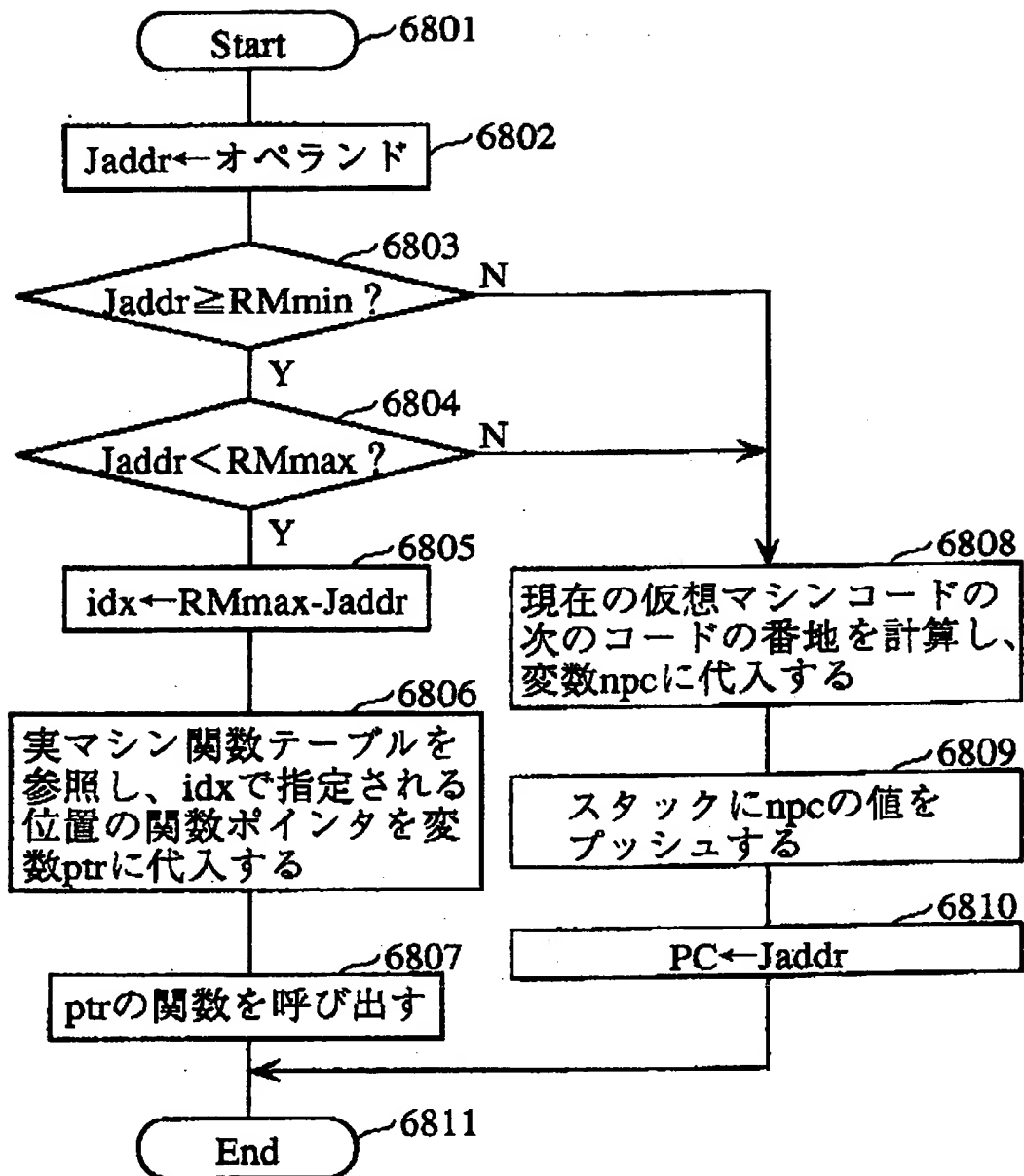
【図 3 4】



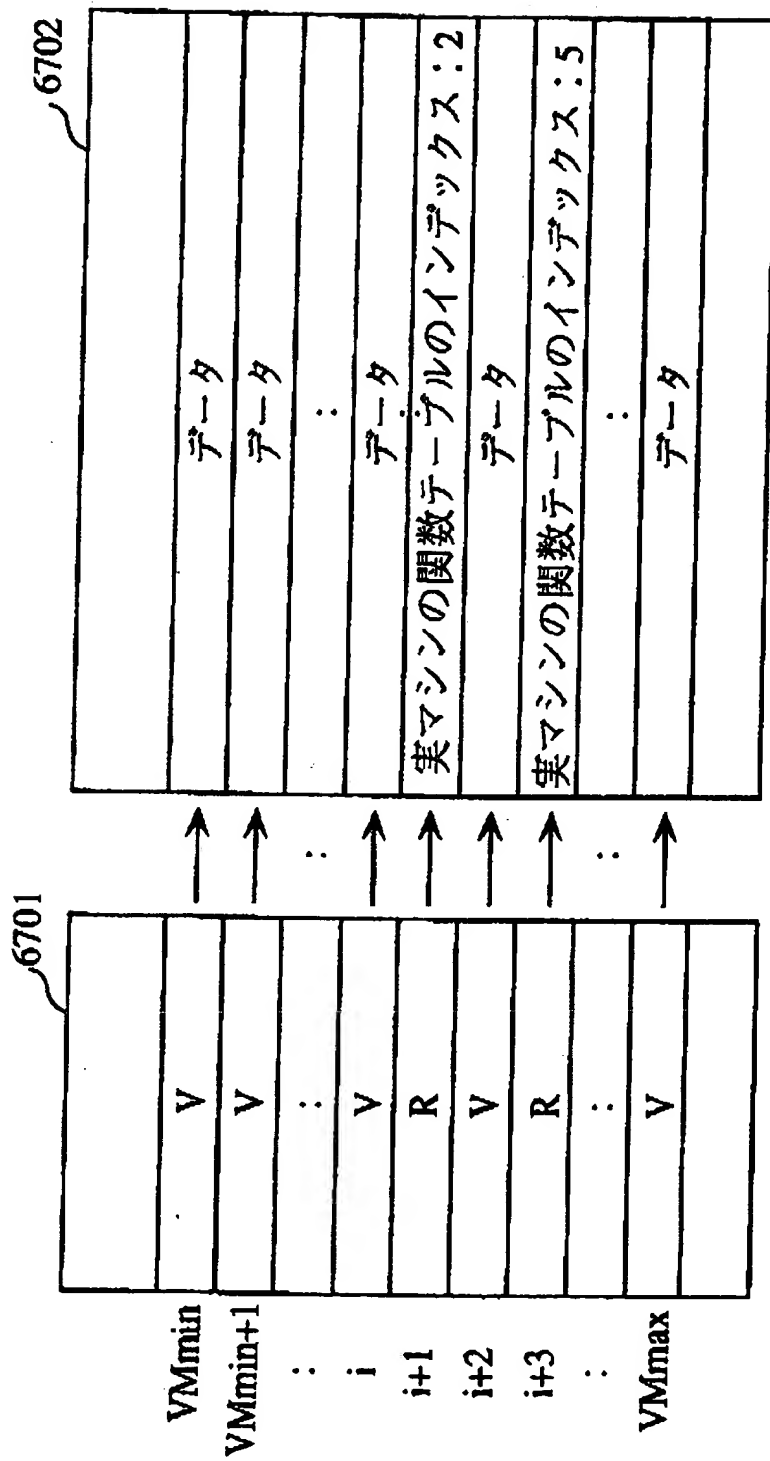
【図 3 5】

(対応する アドレス)	
RMmax	(RMmax-RMmin番の)実マシンでの関数ポインタ
RMmax-1	(RMmax-RMmin-1番の)実マシンでの関数ポインタ
RMmax-2	(RMmax-RMmin-2番の)実マシンでの関数ポインタ
	⋮
RMmin	(0番の)実マシンでの関数ポインタ

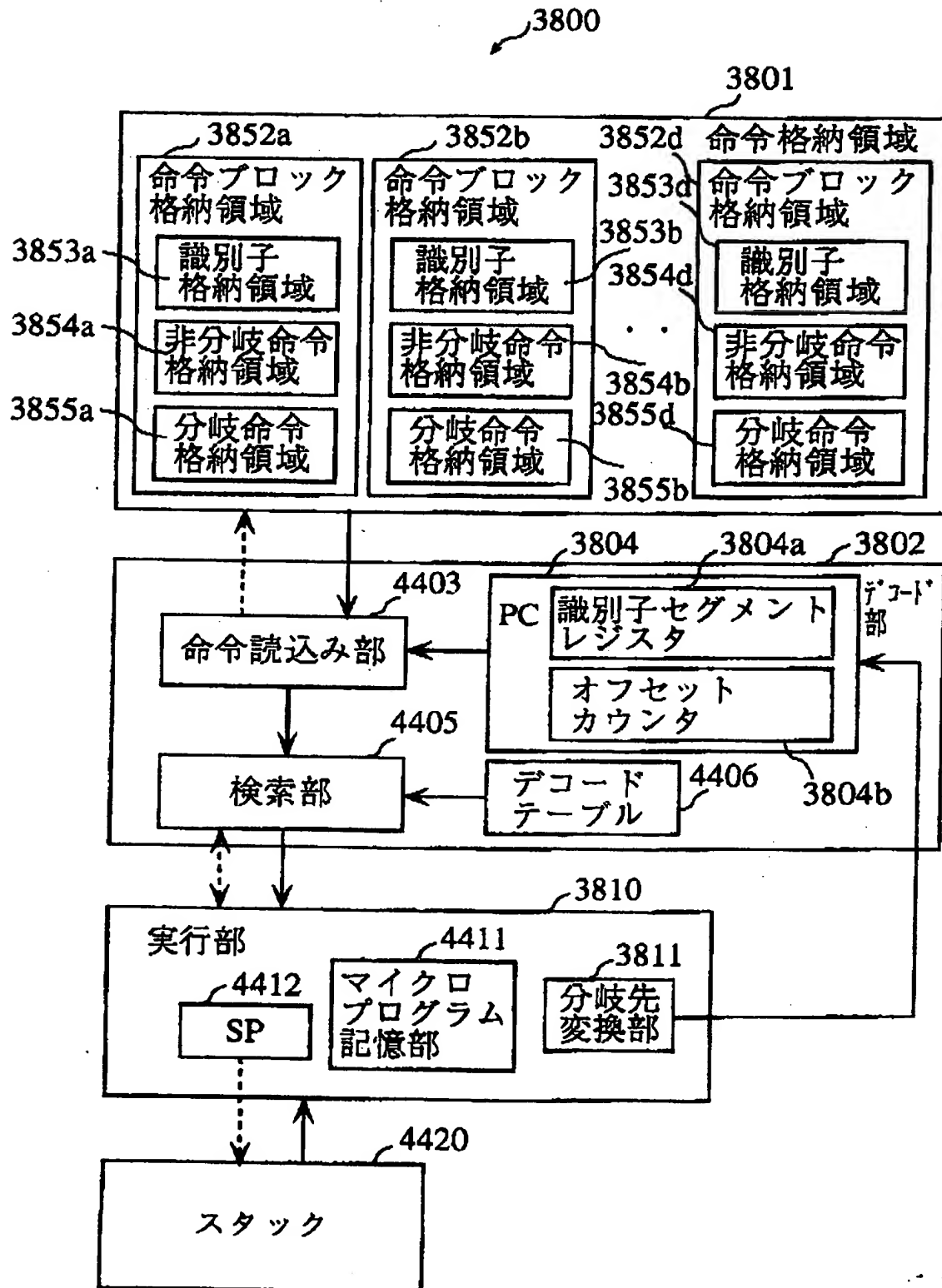
【図 36】



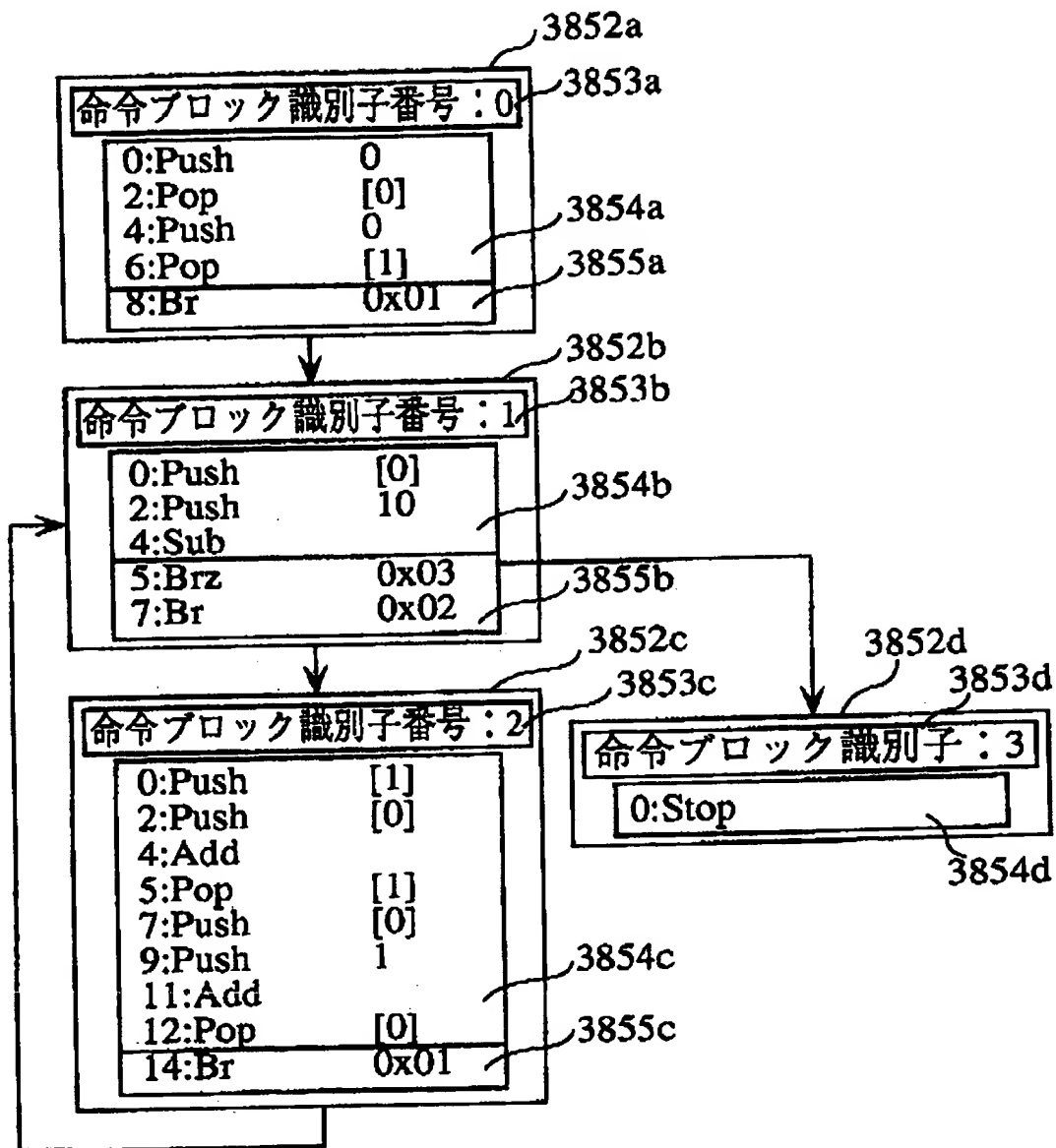
【図 37】



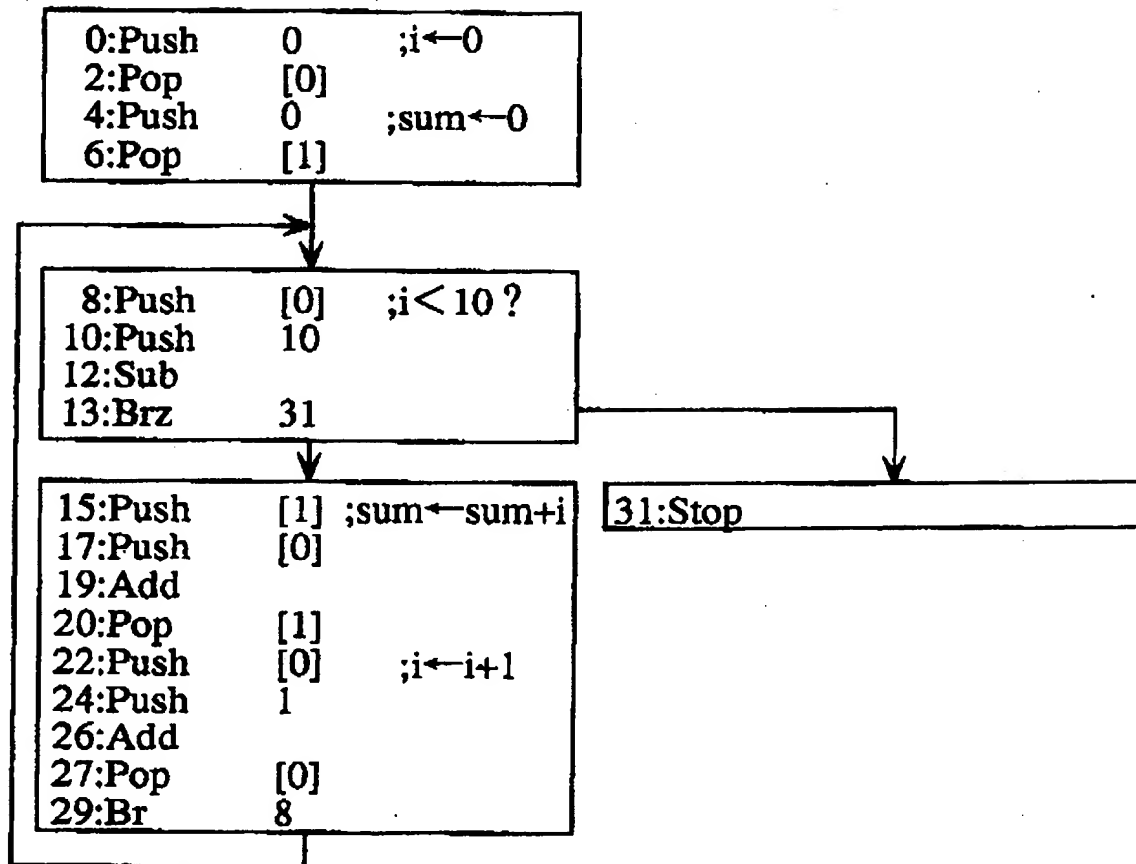
【図 38】



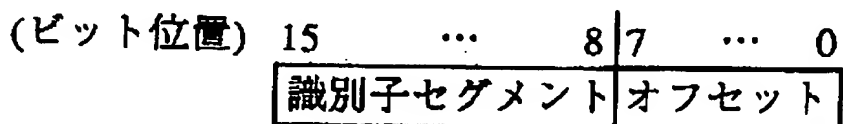
【図 39】



【図 40】

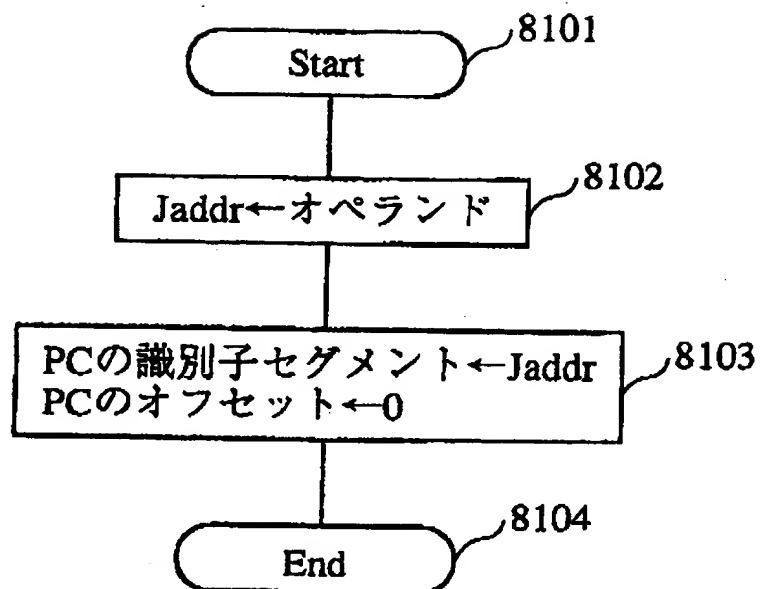


【図 41】

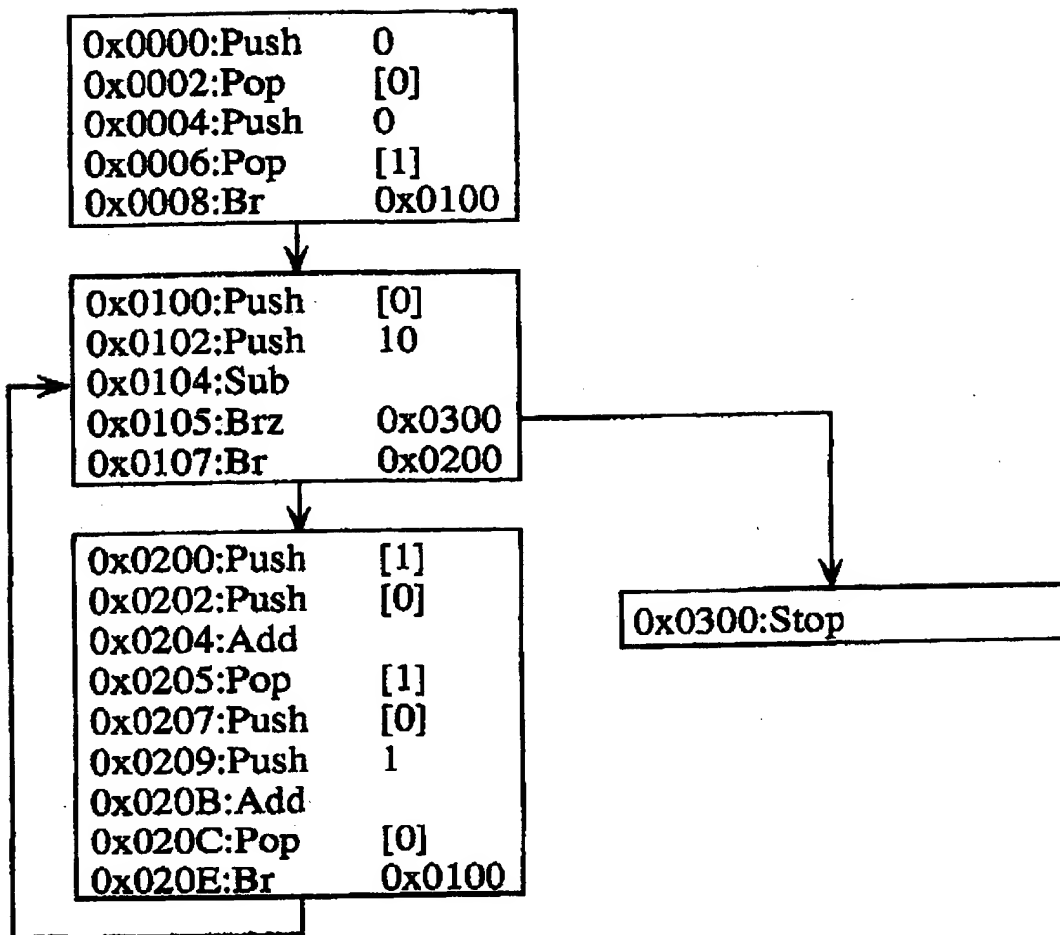




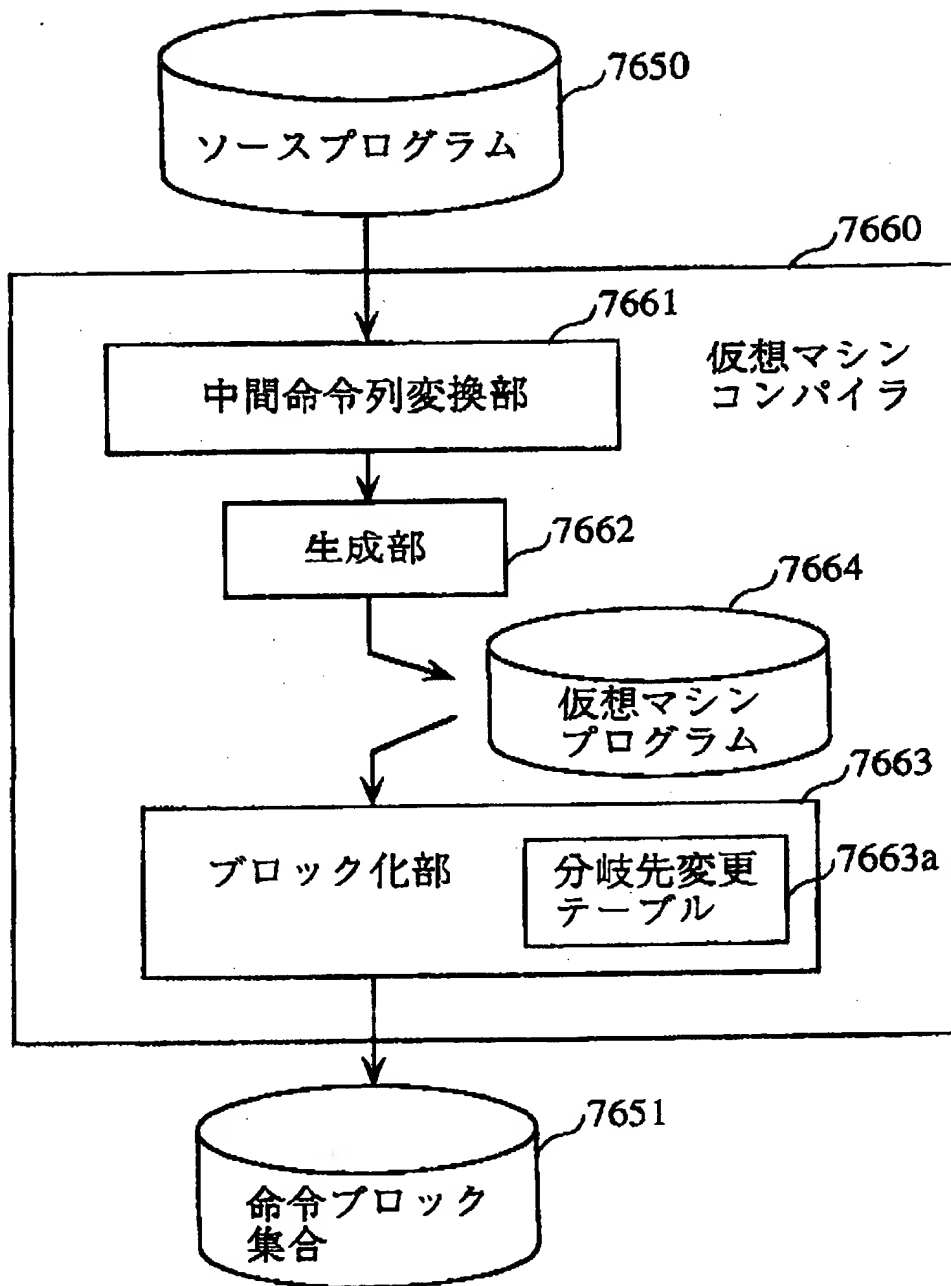
【図 4 2】



【図 4 3】



【図 4 4】

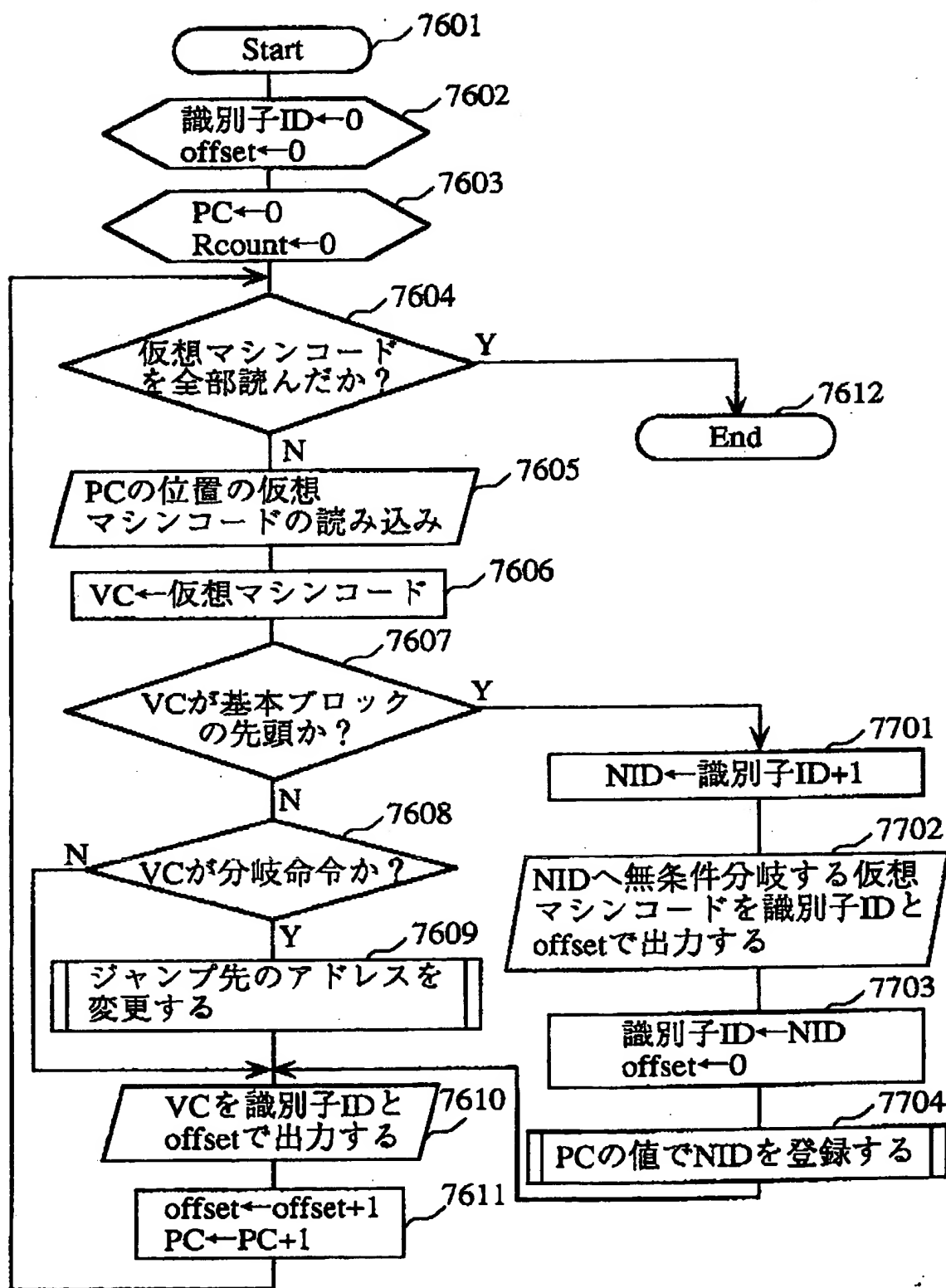


【図 45】

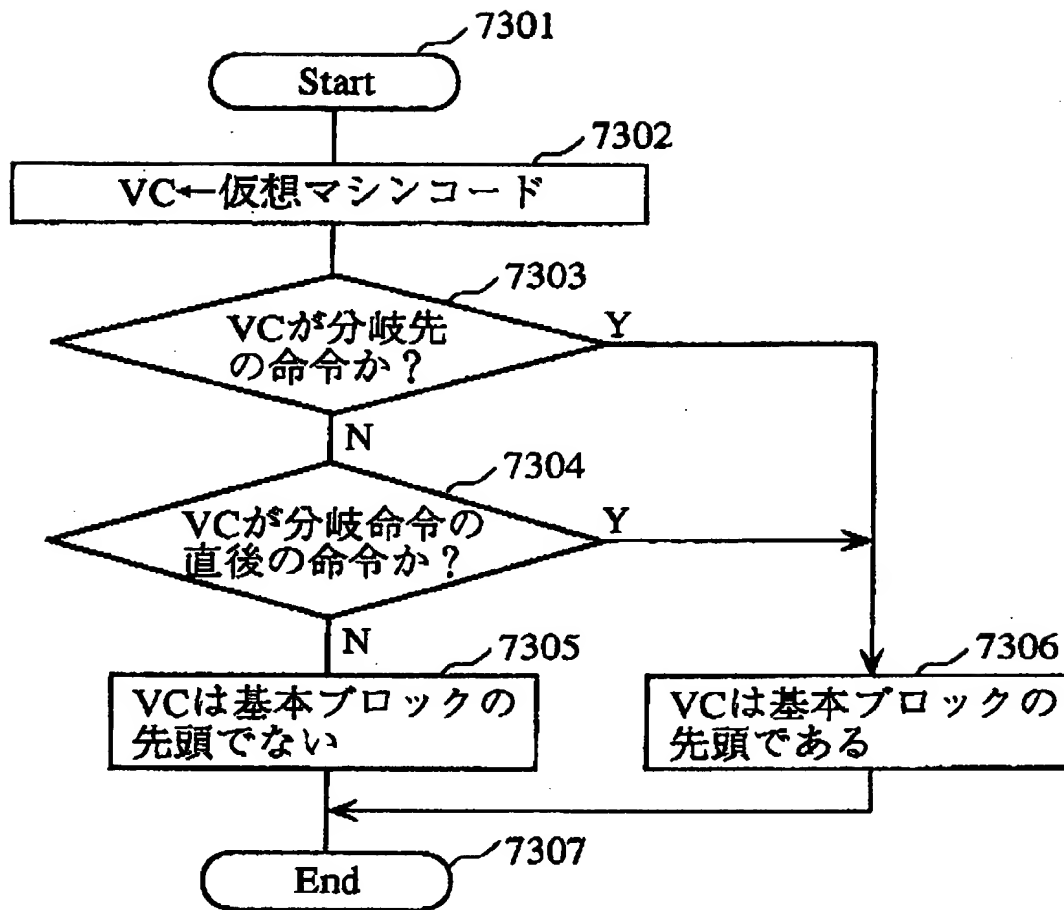
7663a

0	コード位置：	登録フラグ：	参照位置offset：	参照位置識別子：
1	コード位置：	登録フラグ：	参照位置offset：	参照位置識別子：
2	コード位置：	登録フラグ：	参照位置offset：	参照位置識別子：
3	コード位置：	登録フラグ：	参照位置offset：	参照位置識別子：
4	コード位置：	登録フラグ：	参照位置offset：	参照位置識別子：
...				
Rcount-1	コード位置：	登録フラグ：	参照位置offset：	参照位置識別子：

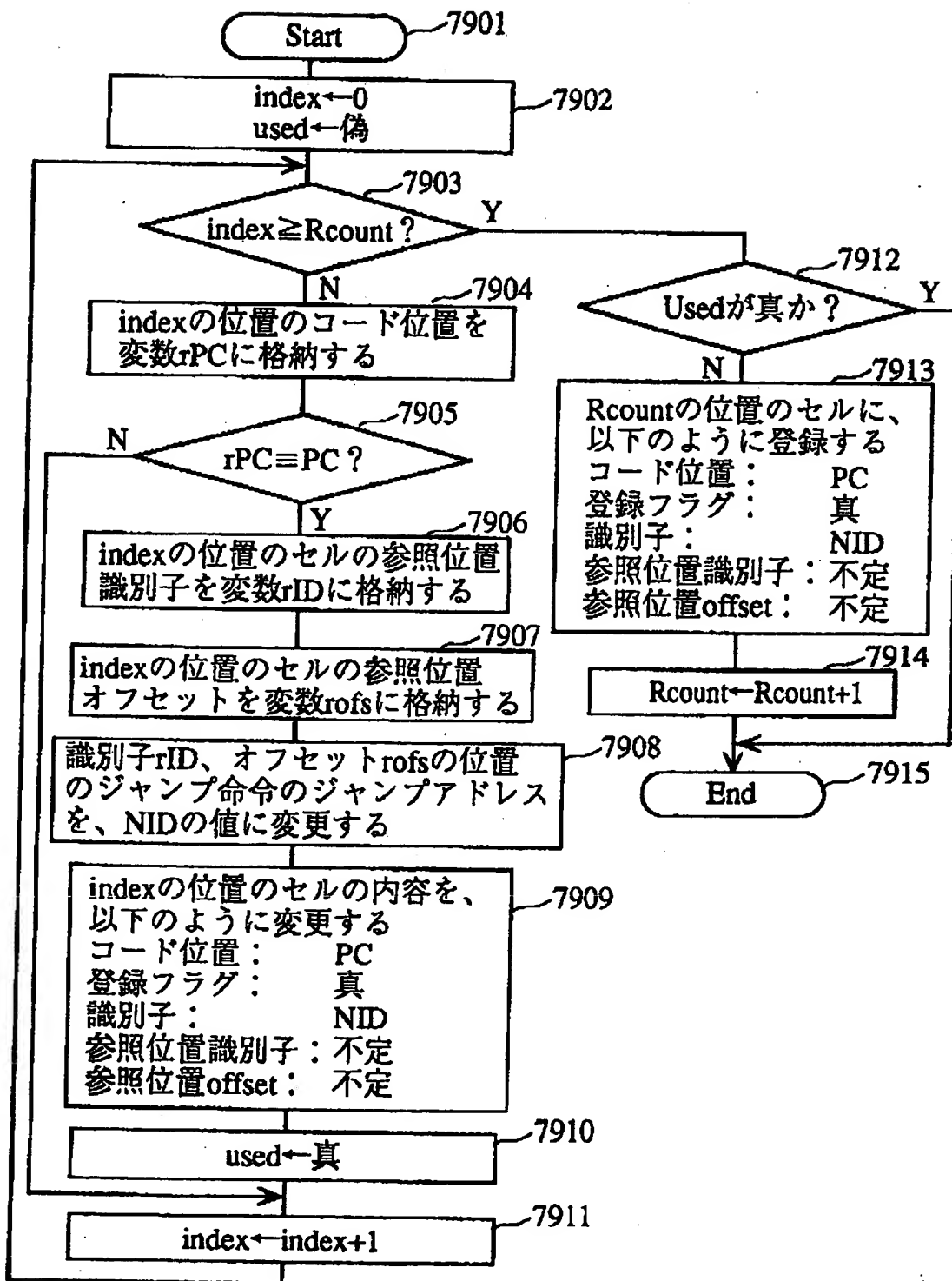
【図 4 6】



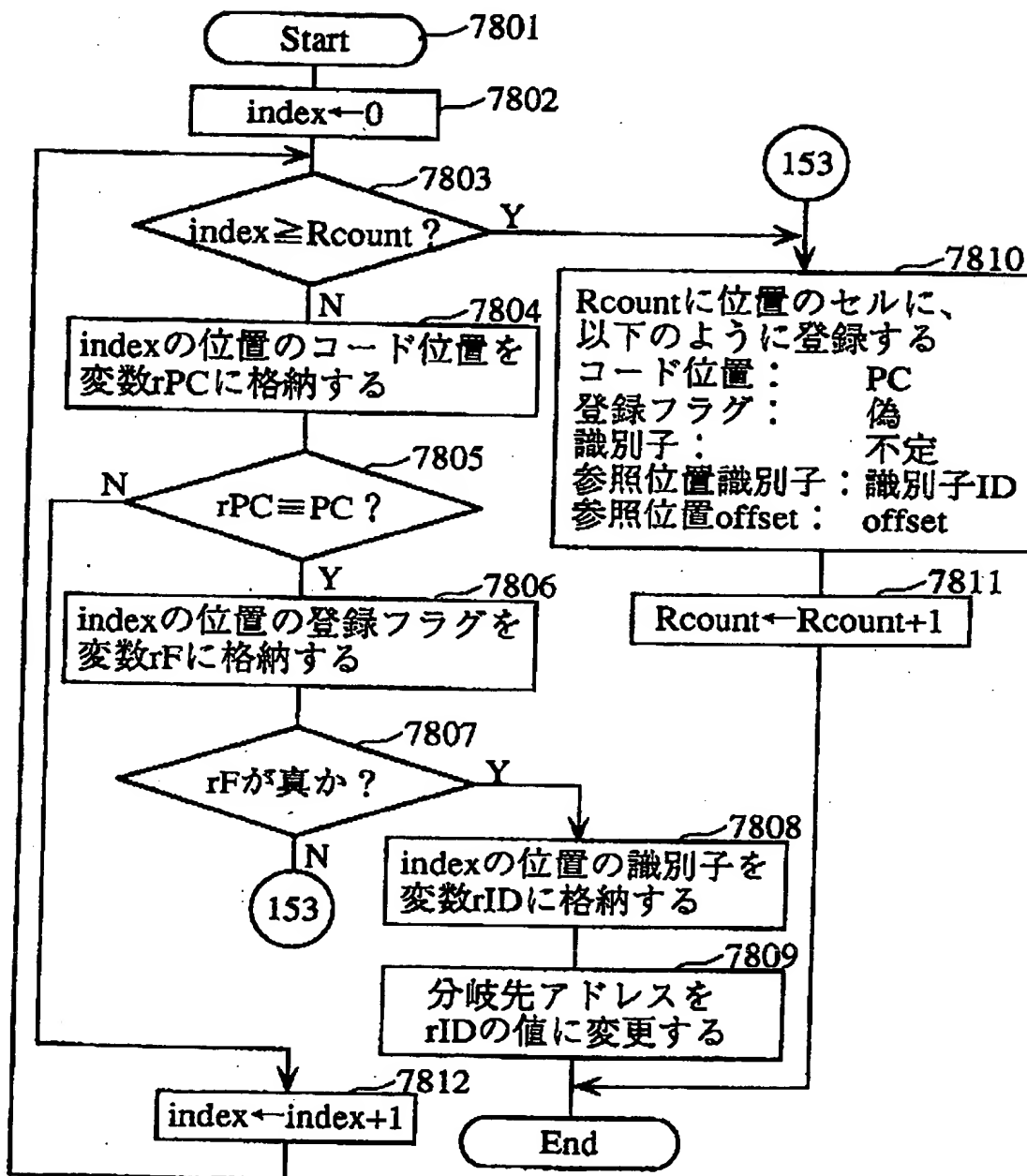
【図 47】



【図 48】

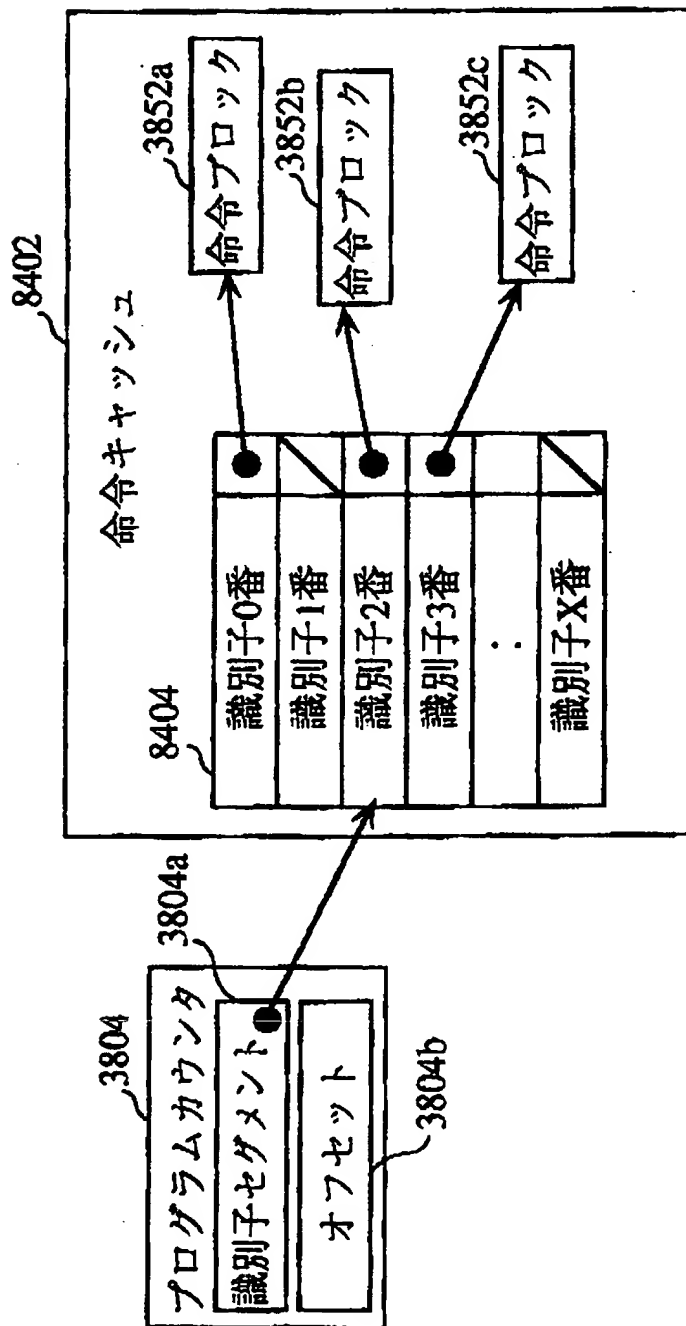


【図 49】

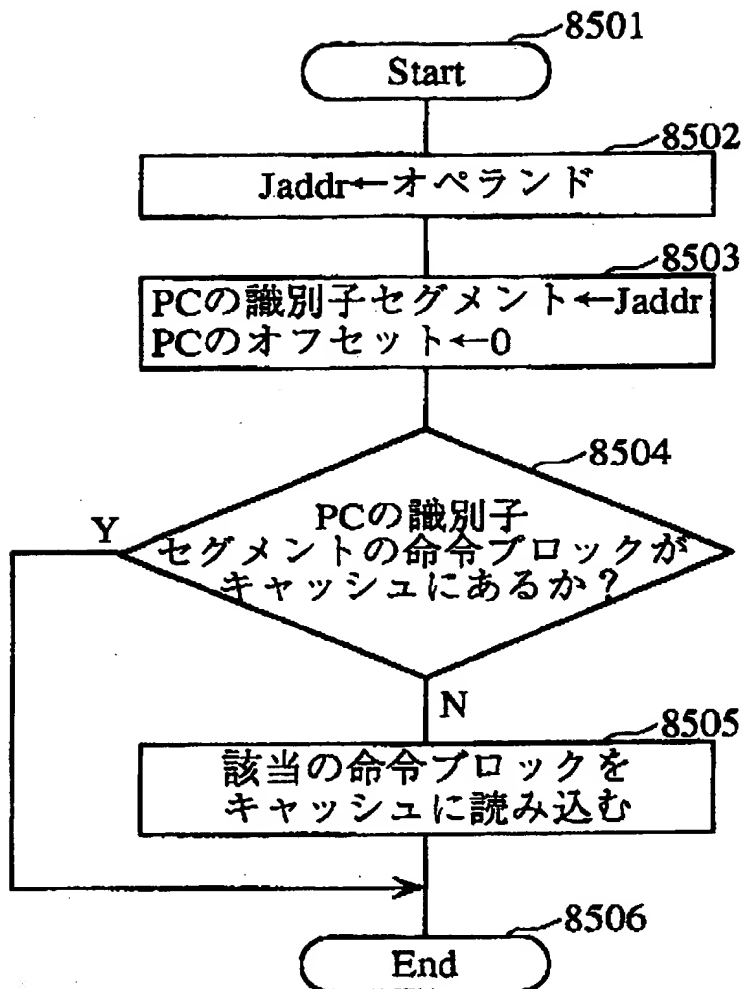




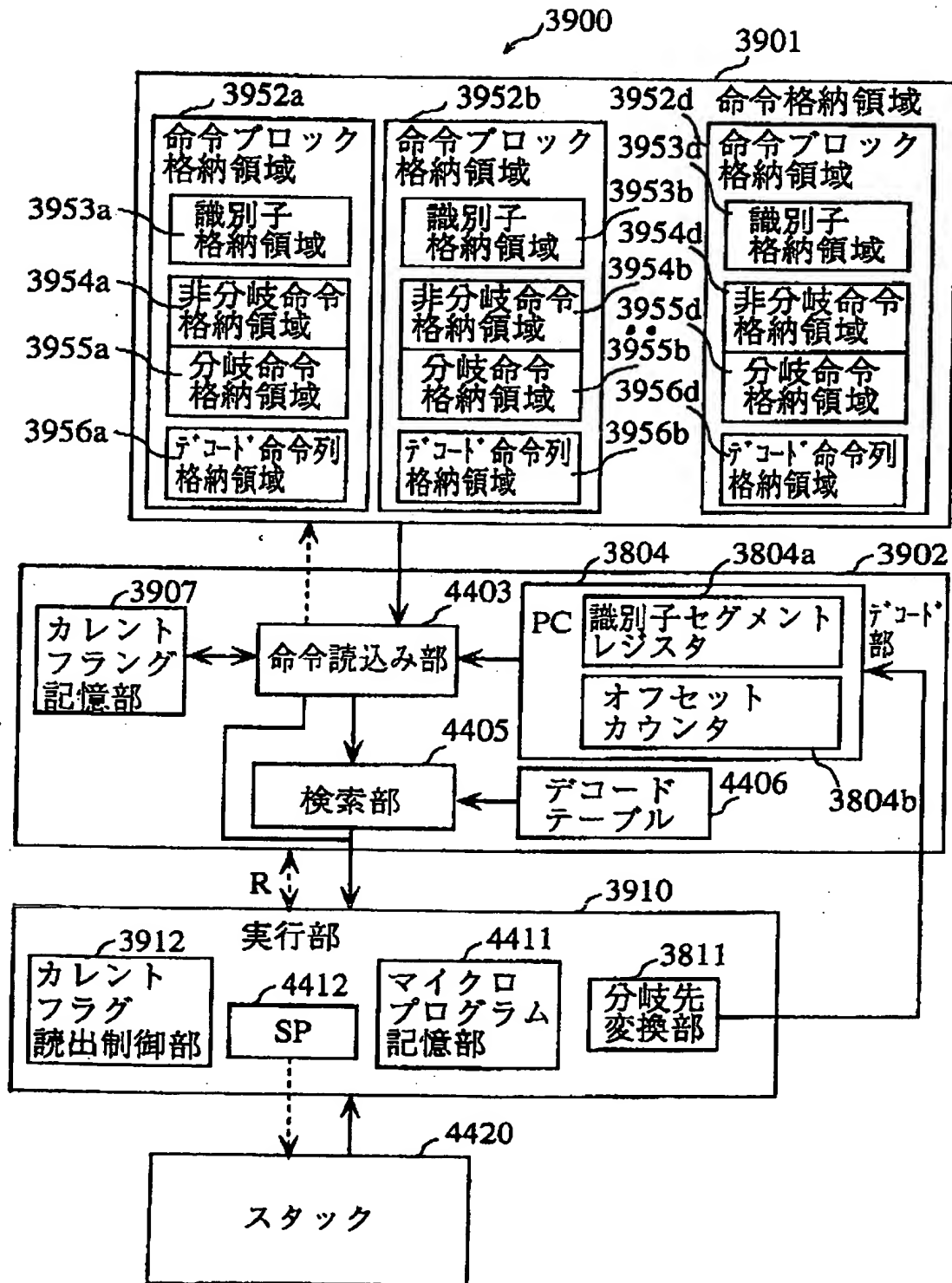
【図 50】



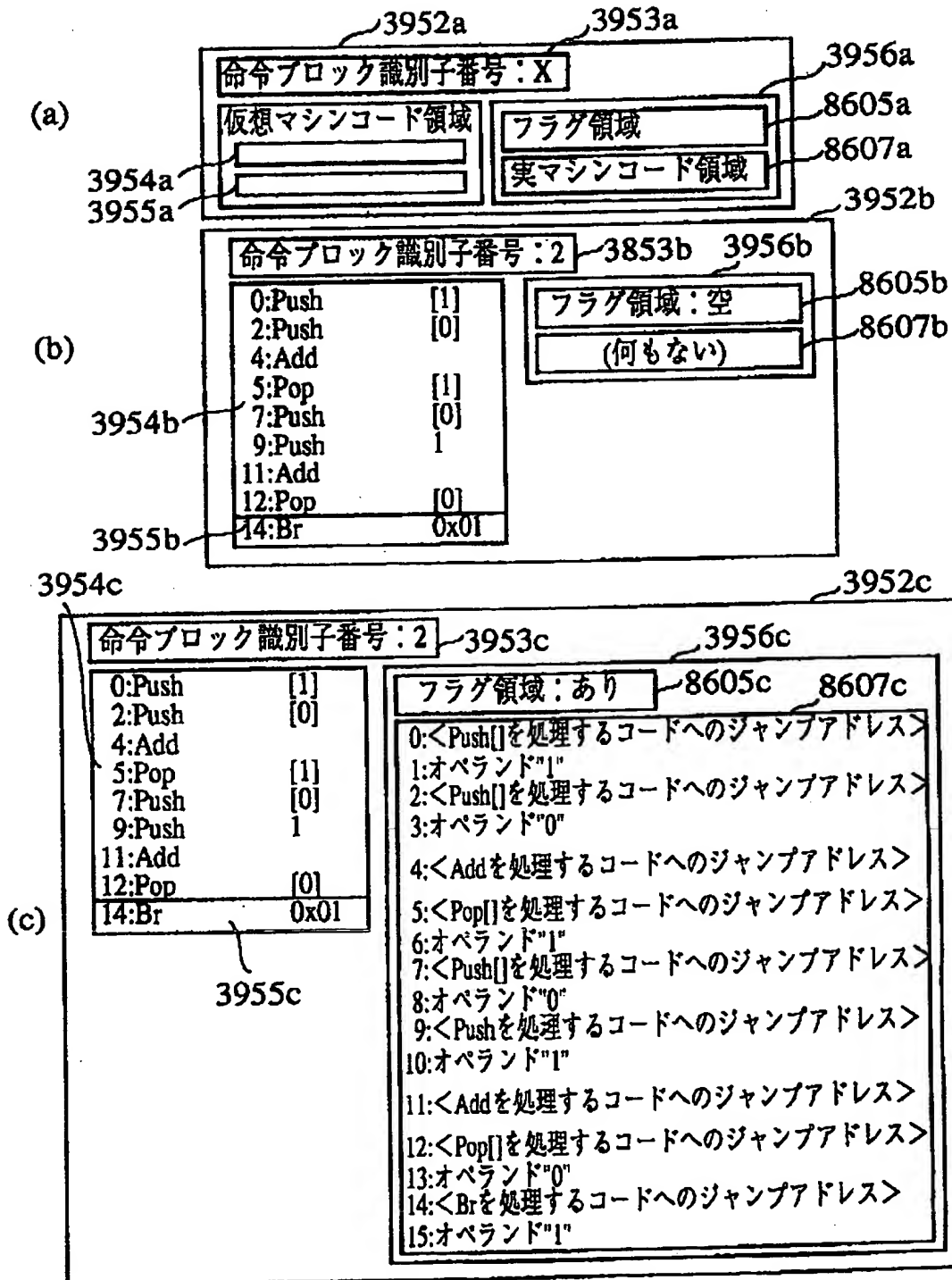
【図 51】



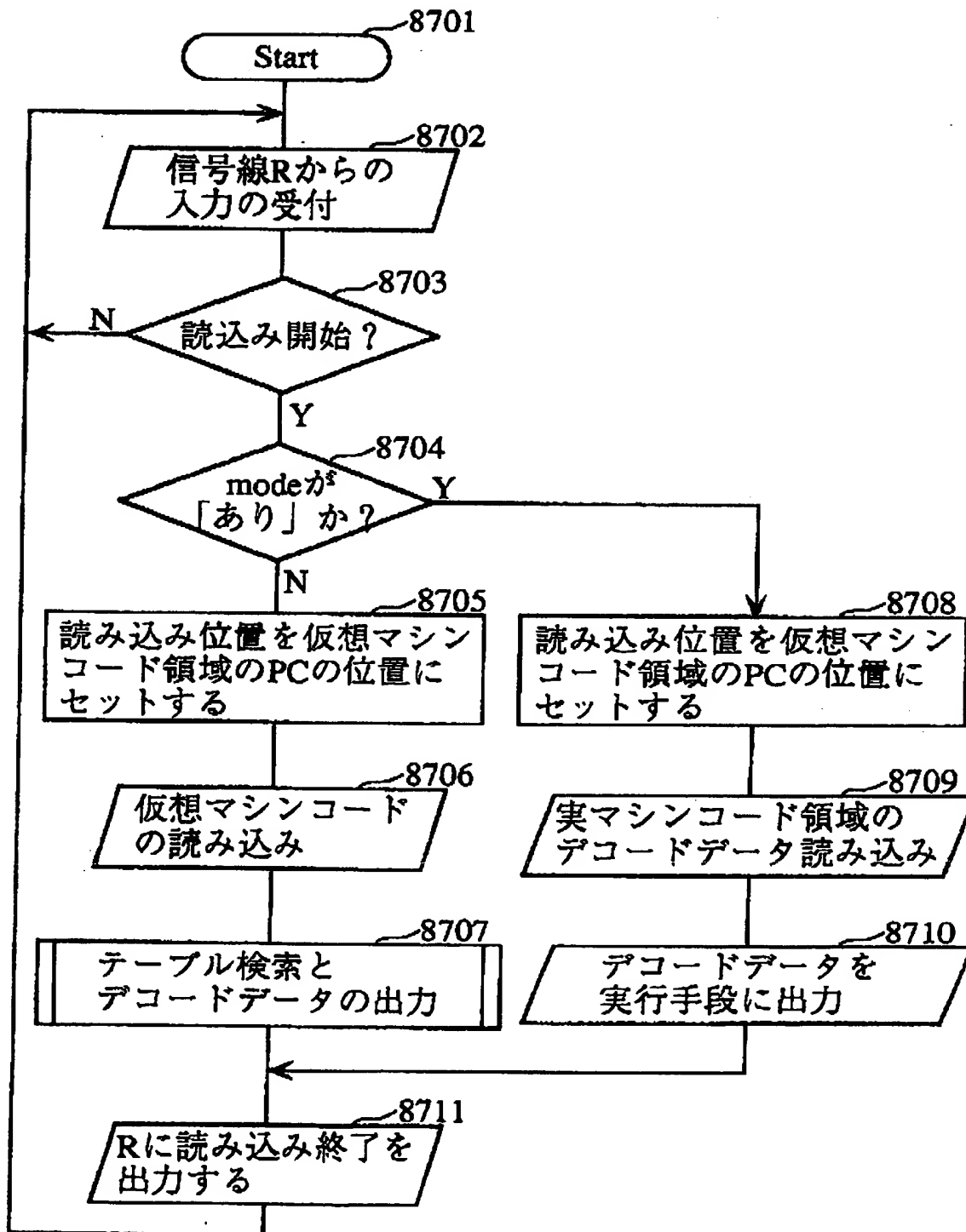
【図 5 2】



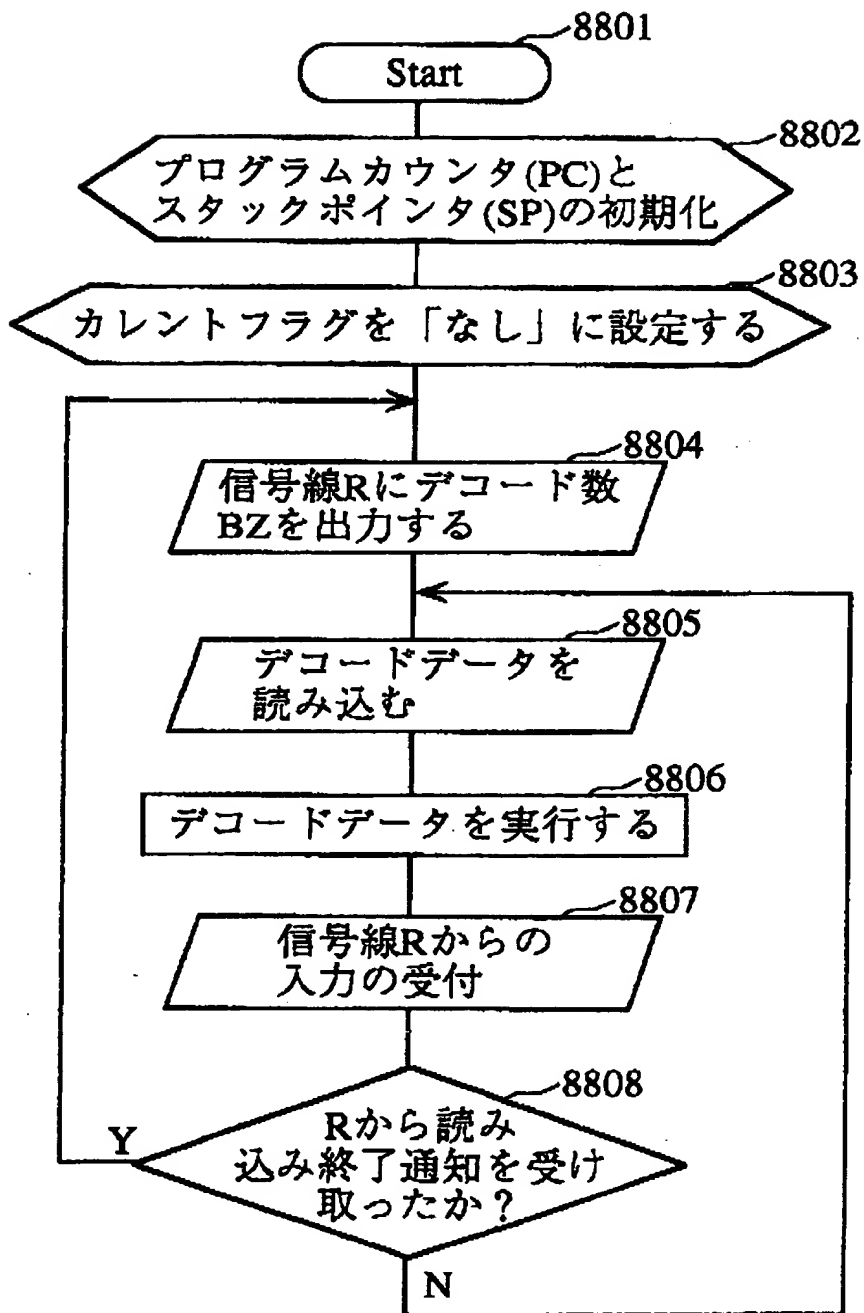
【図 53】



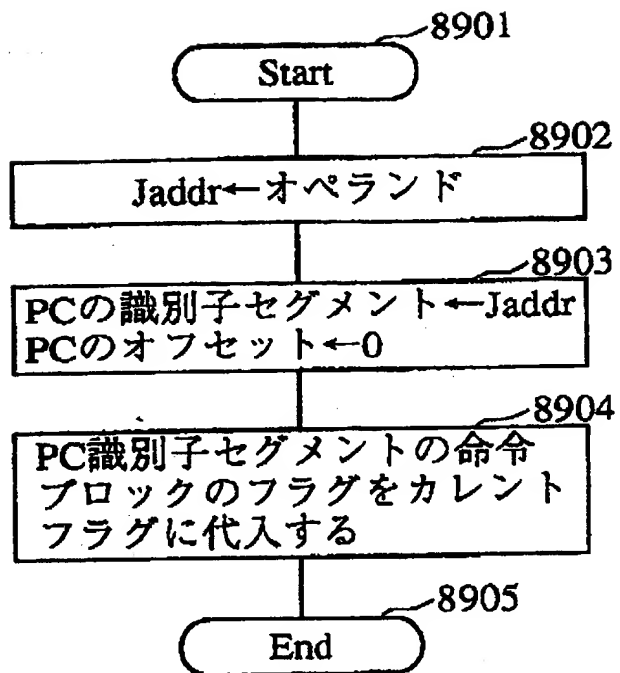
【図 54】



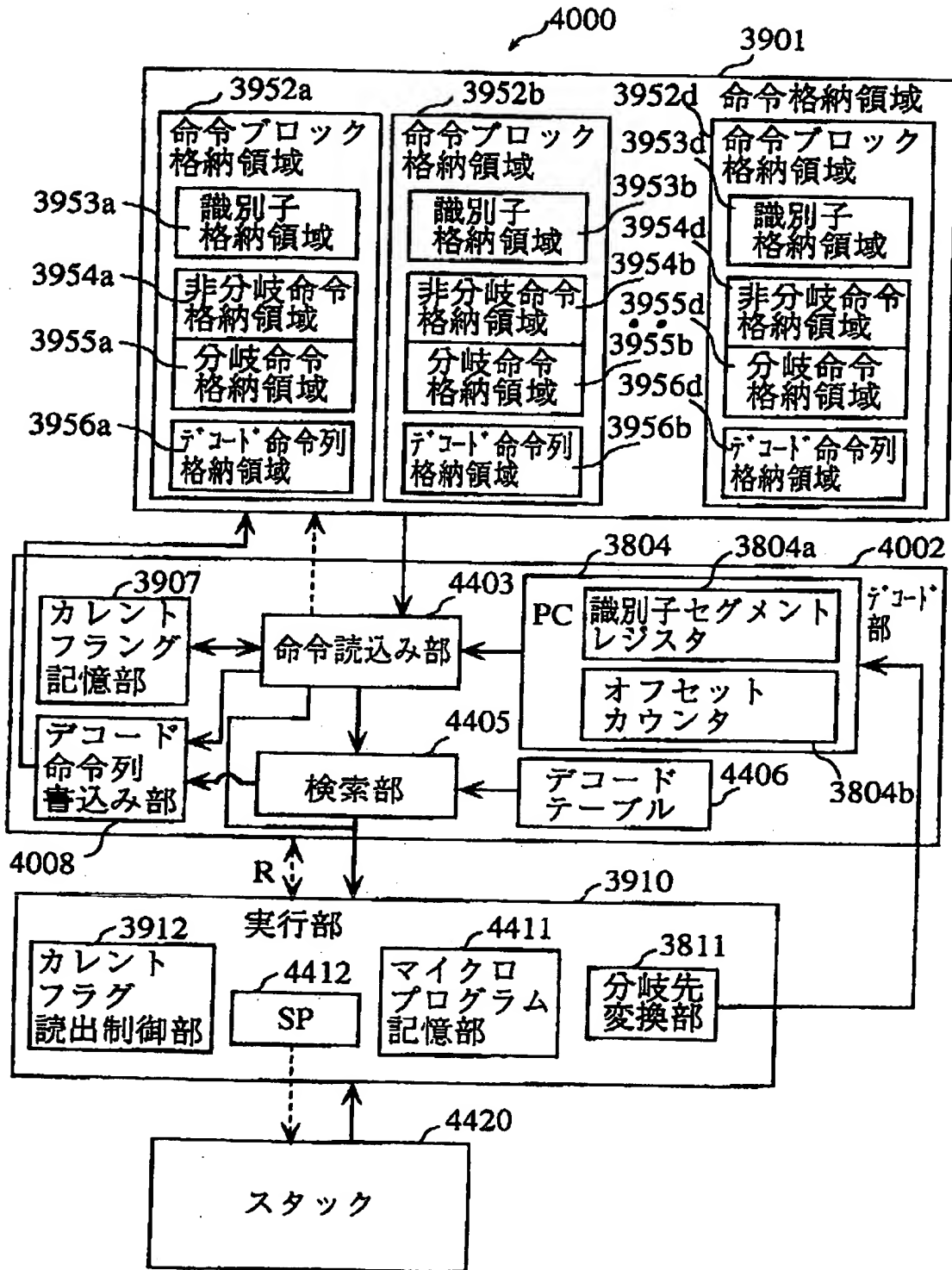
【図 55】



【図 56】

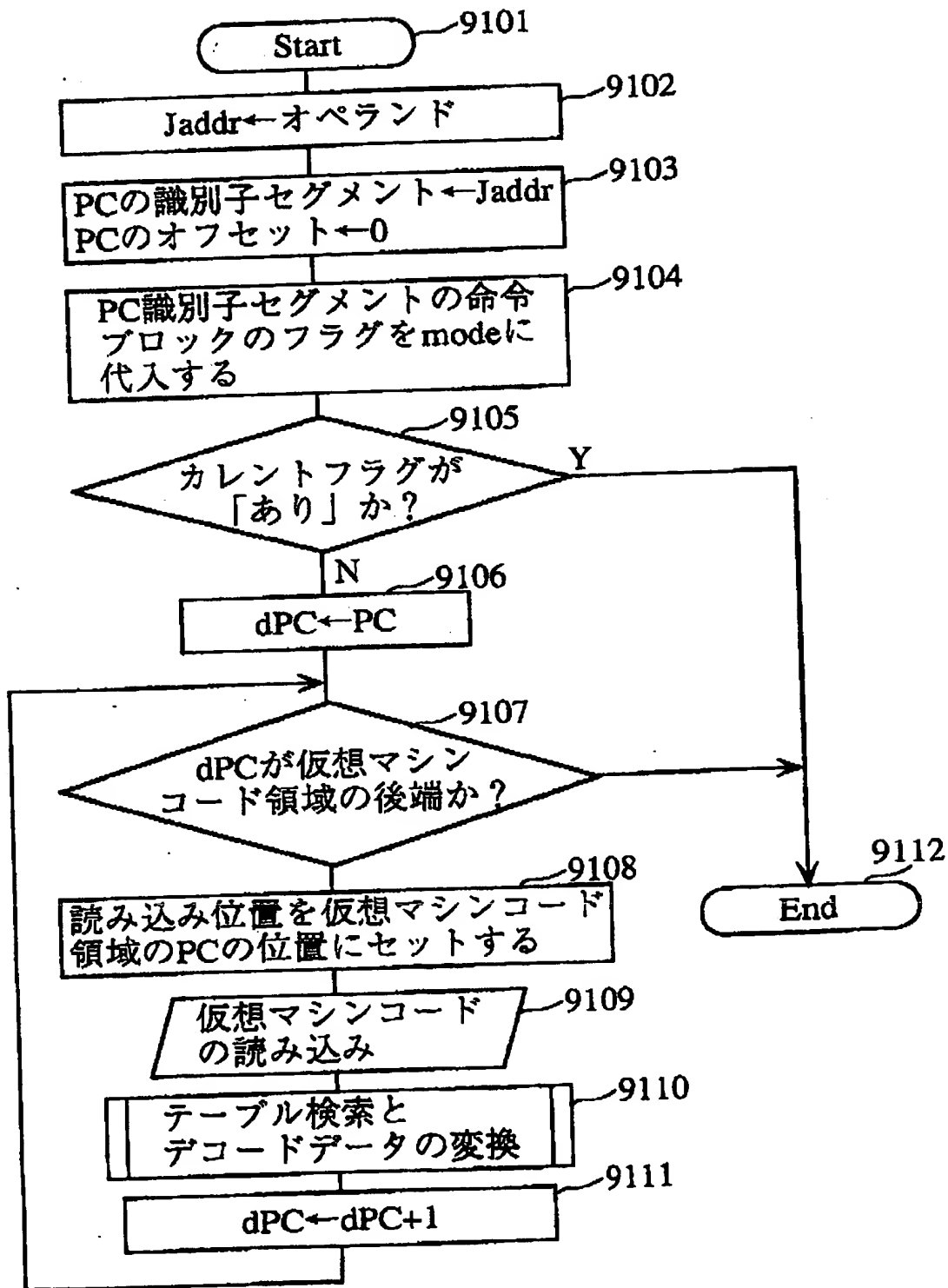


【図 57】

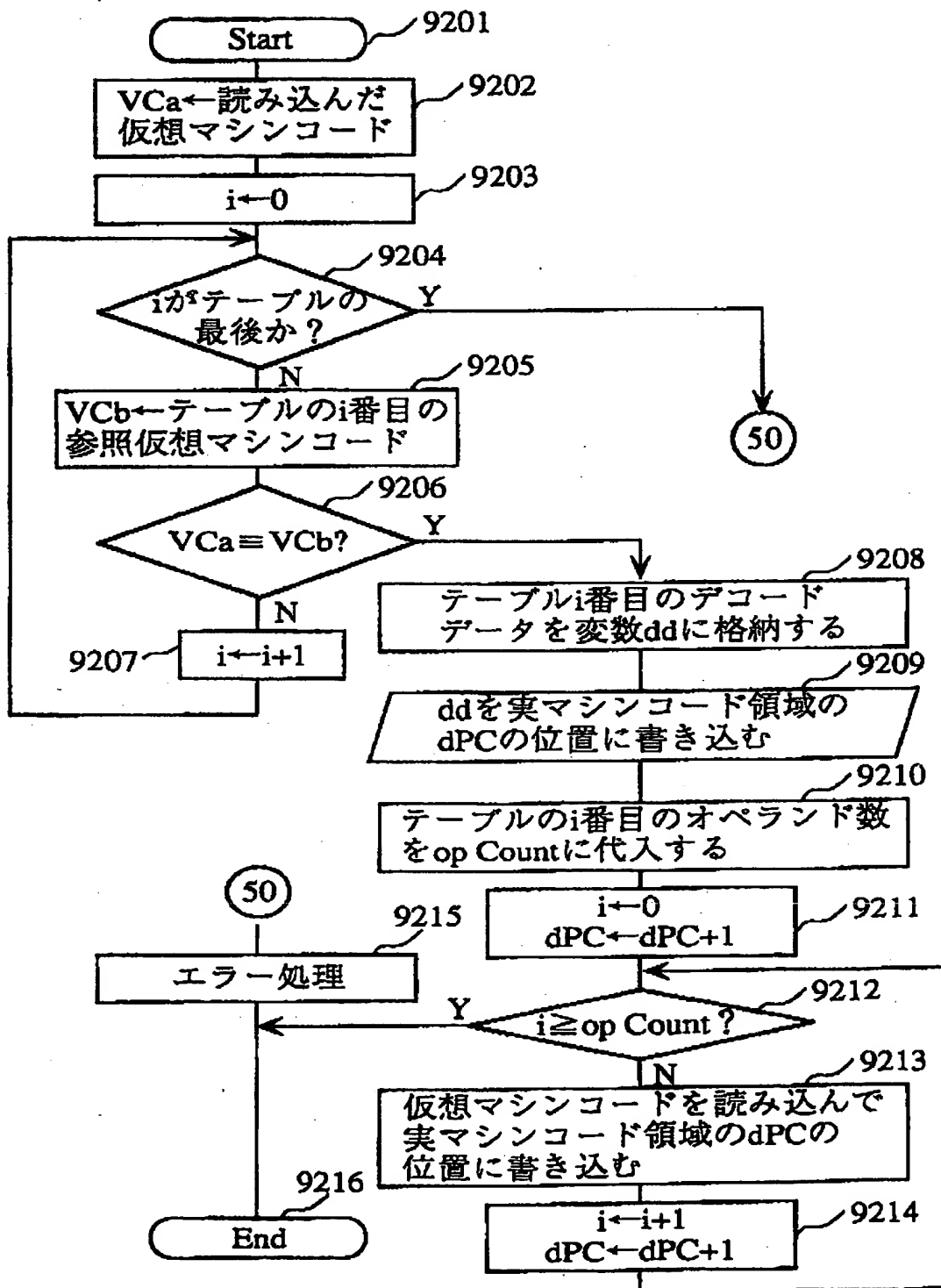




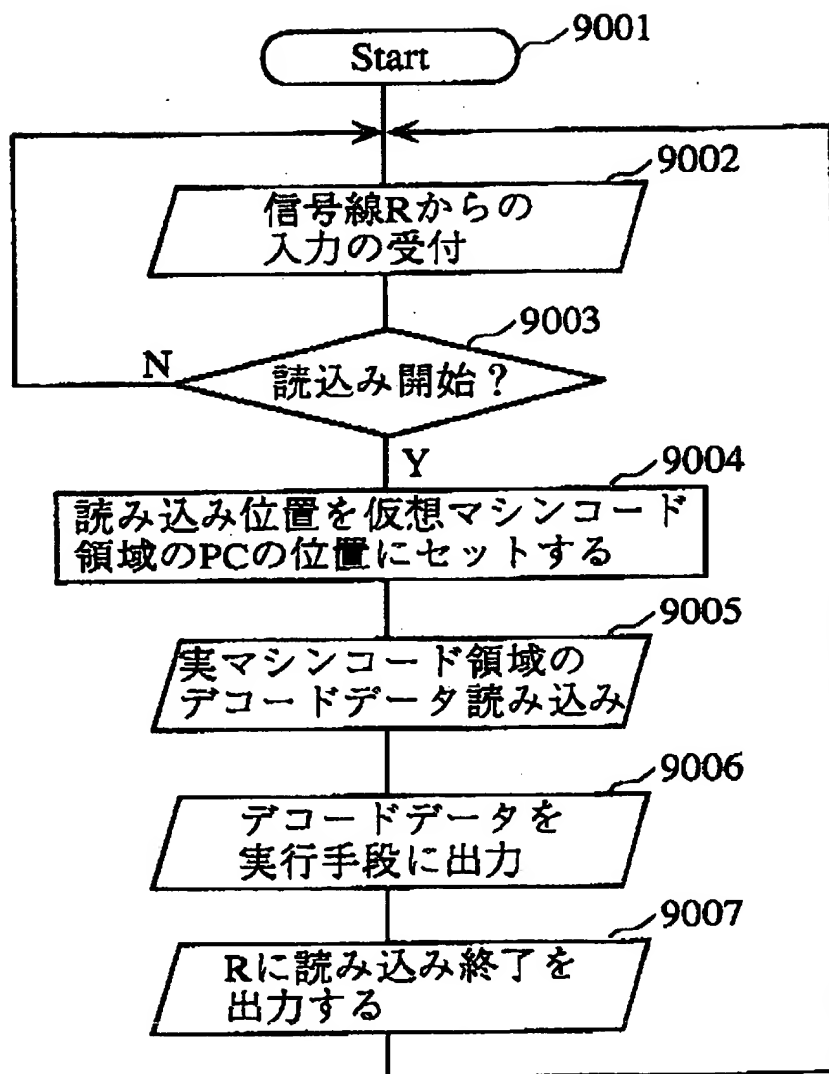
【図 58】



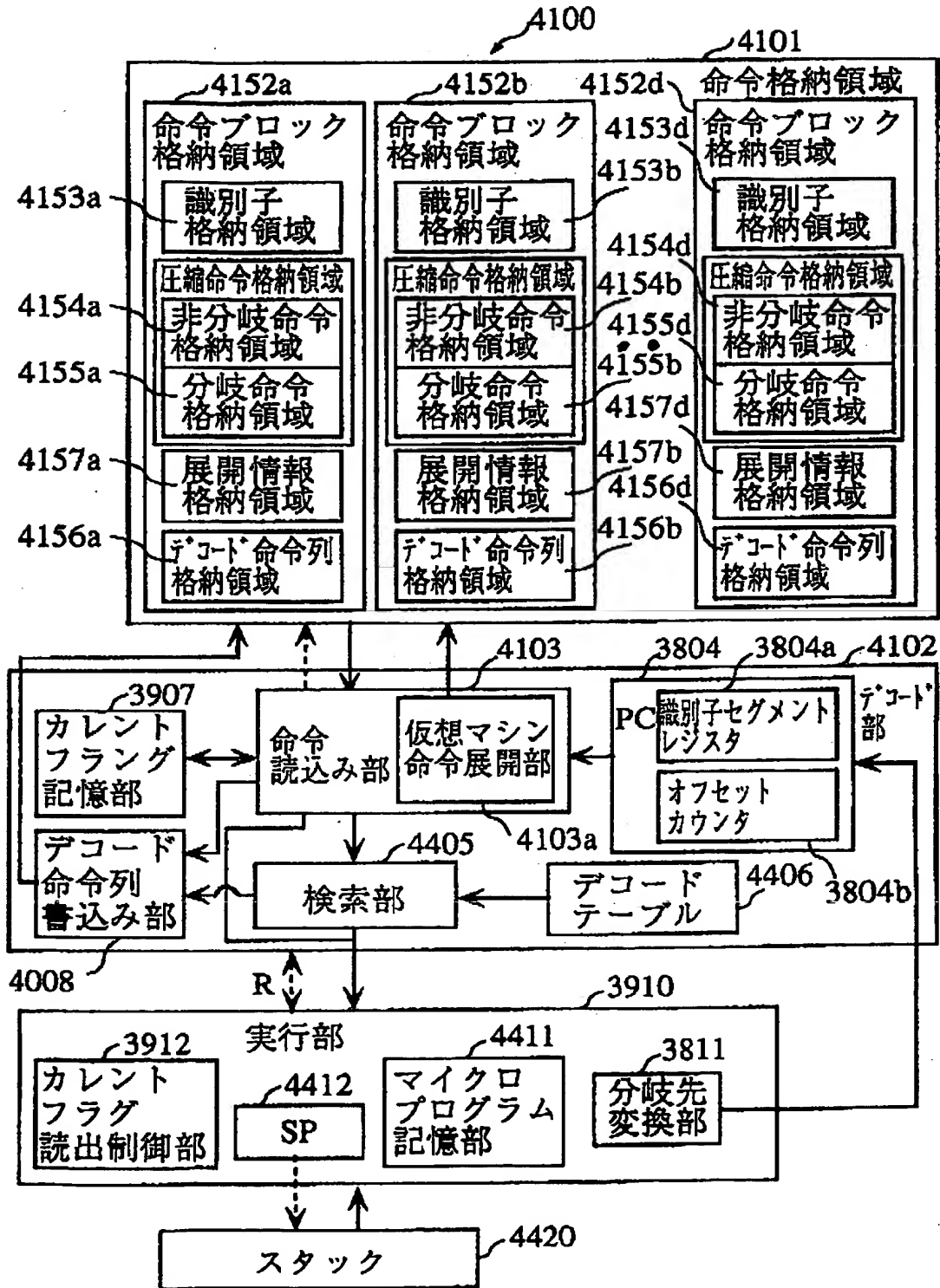
【図 59】



【図 60】



【図 61】

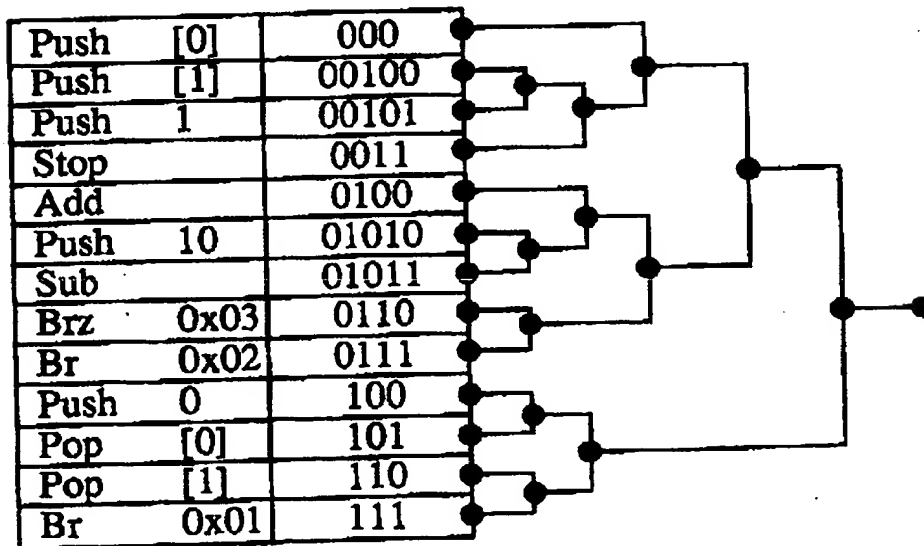


【図 6 2】

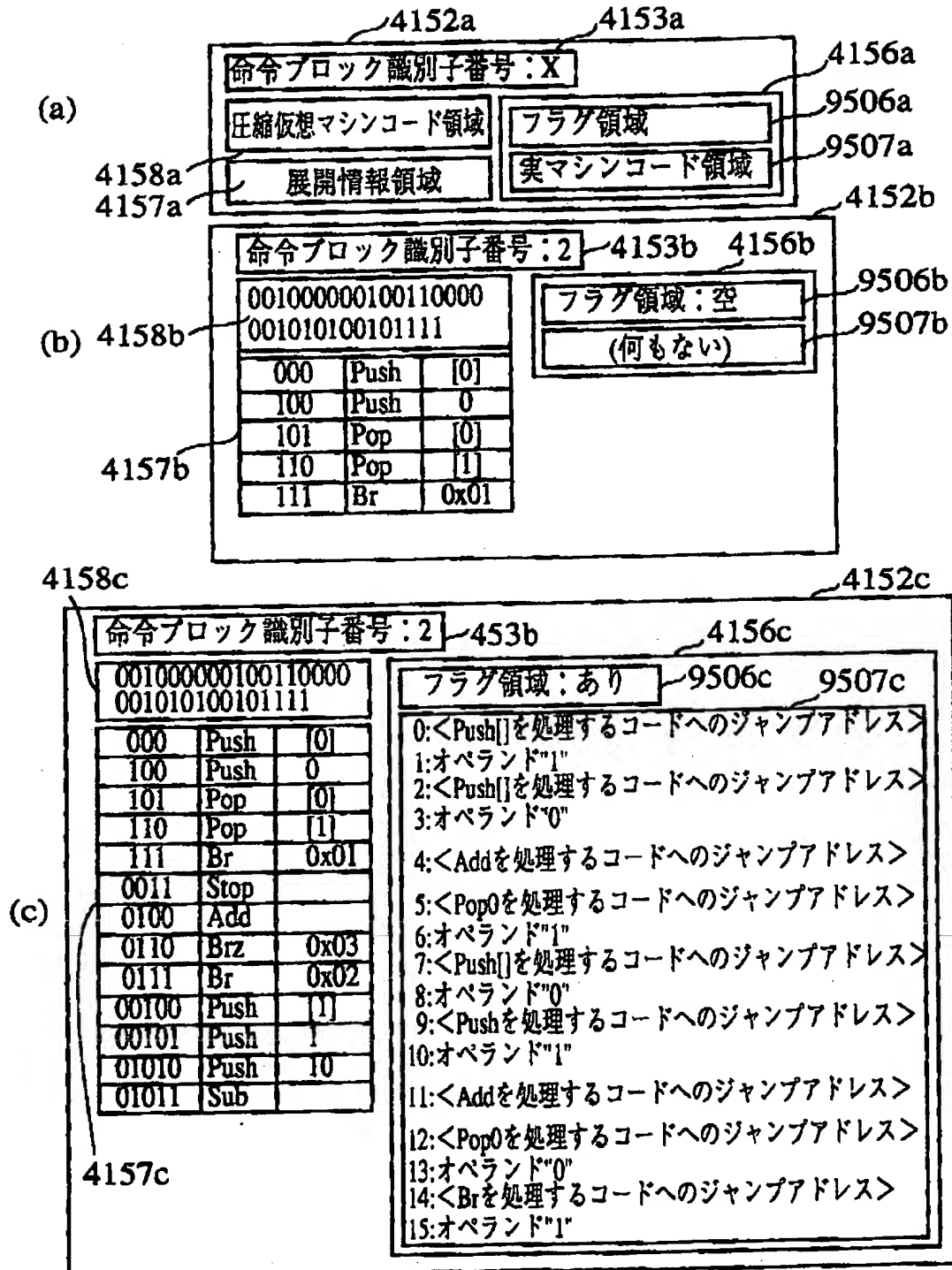
(a)

ビット列	意味
000	Push [0]
100	Push 0
101	Pop [0]
110	Pop [1]
111	Br 0x01
0011	Stop
0100	Add
0110	Brz 0x03
0111	Br 0x02
00100	Push [1]
00101	Push 1
01010	Push 10
01011	Sub

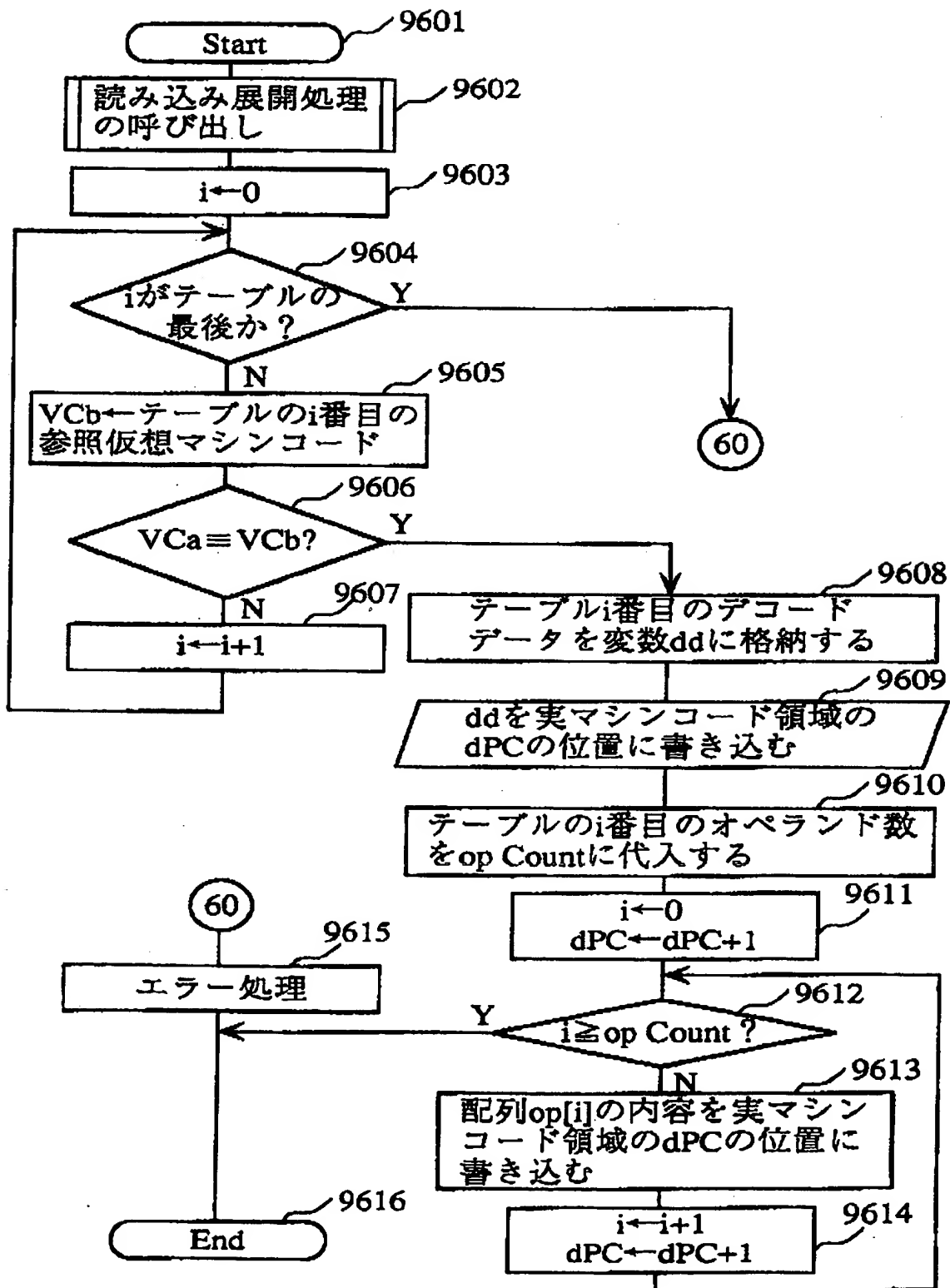
(b)



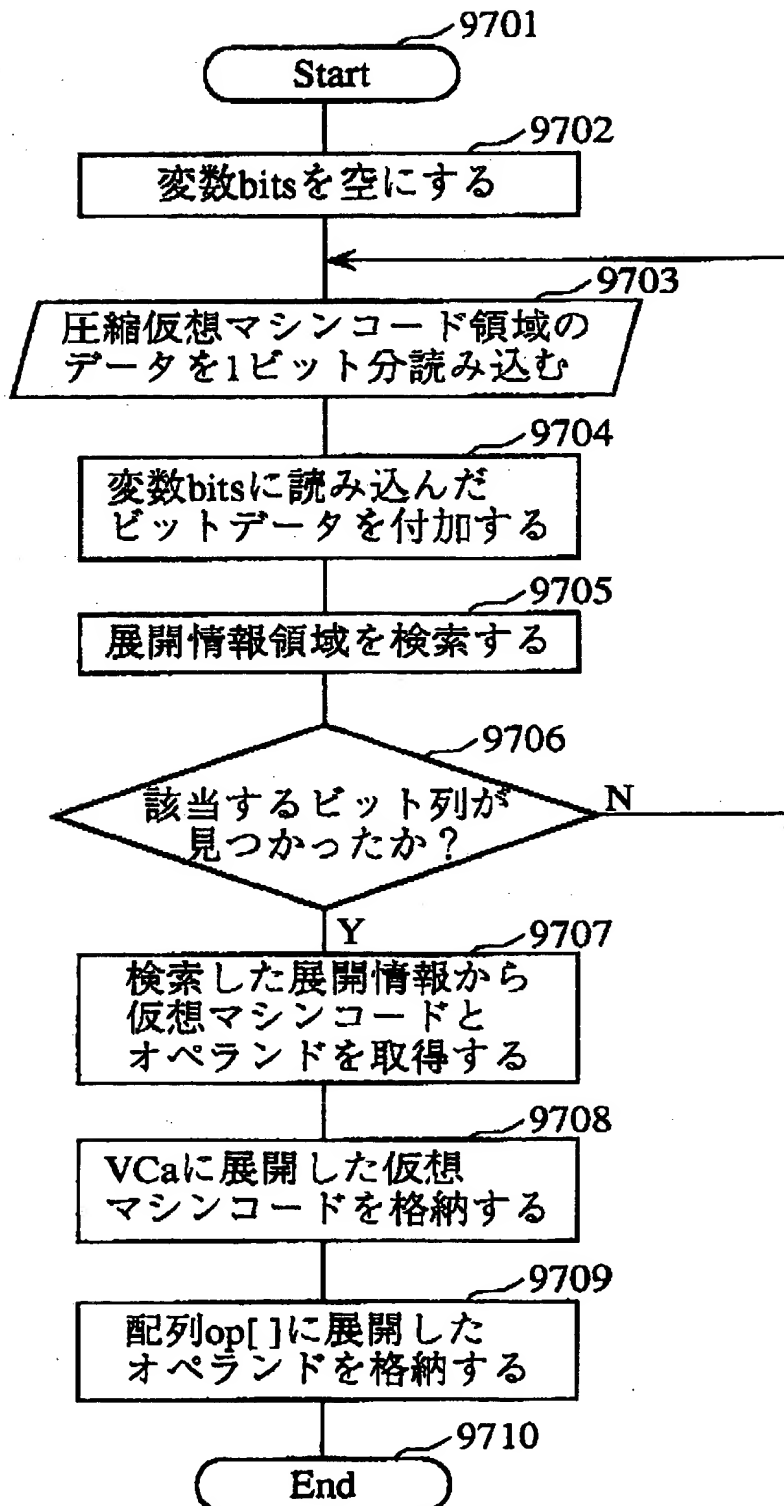
【図 63】



【図 64】

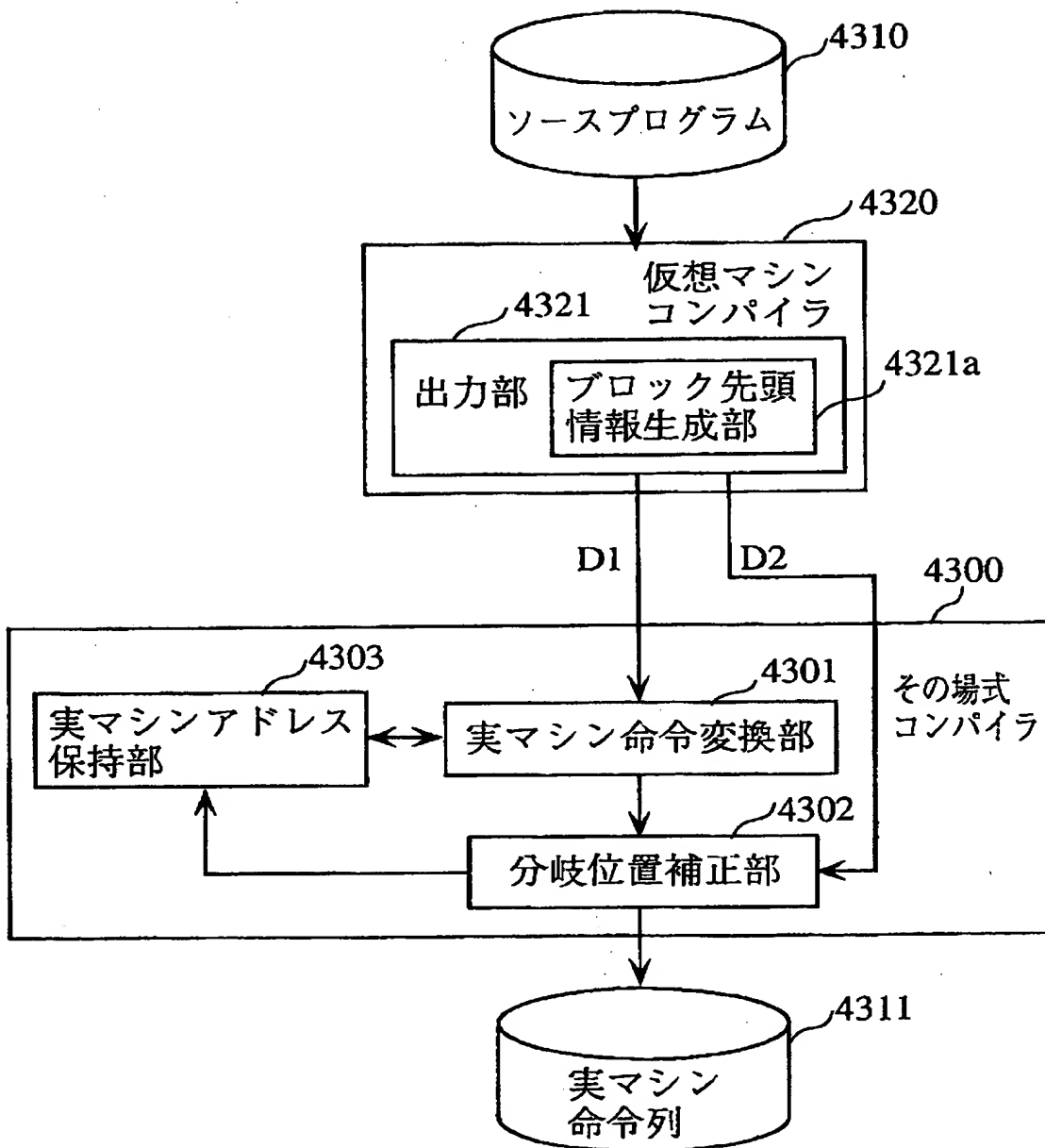


【図 65】

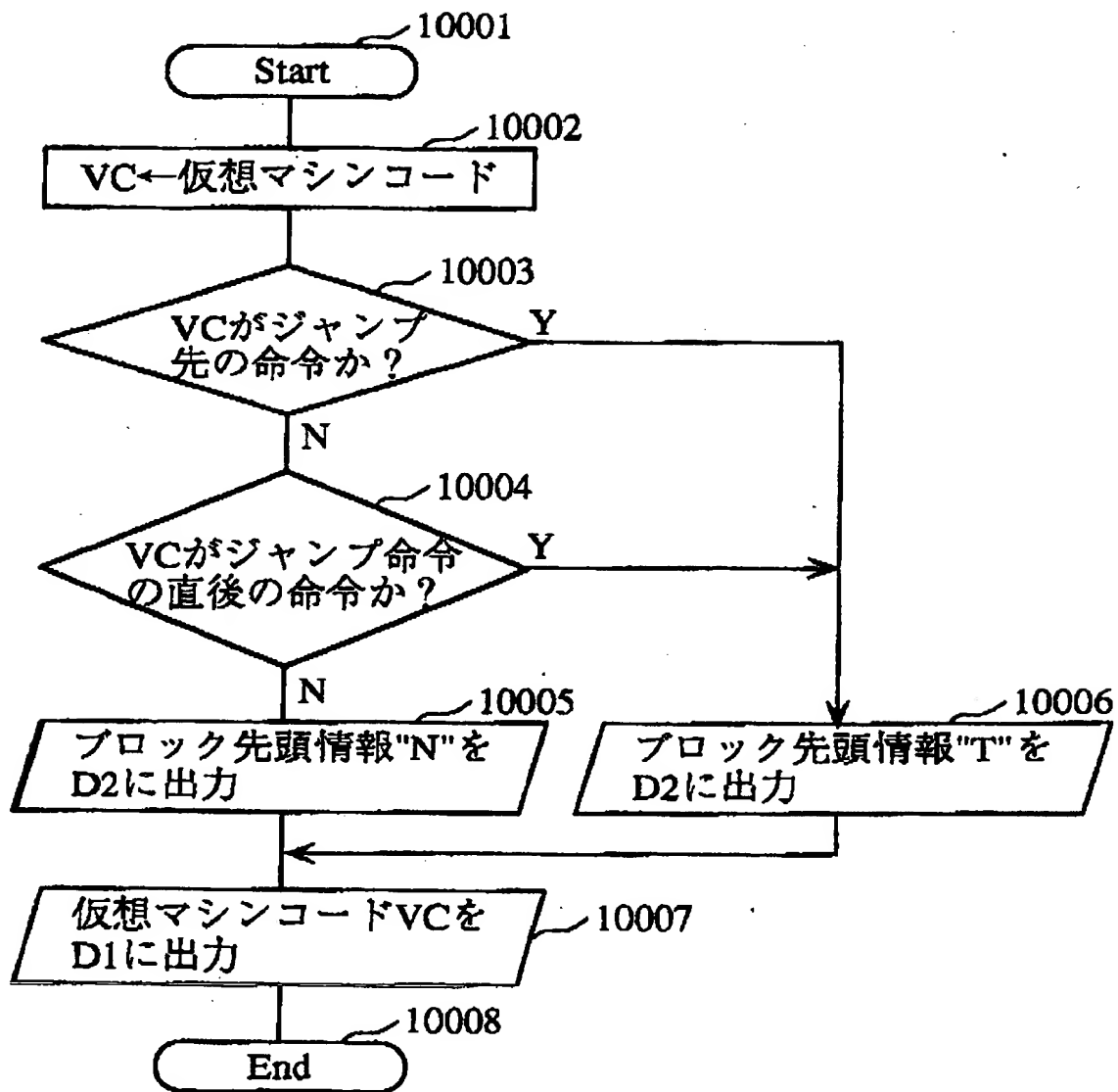




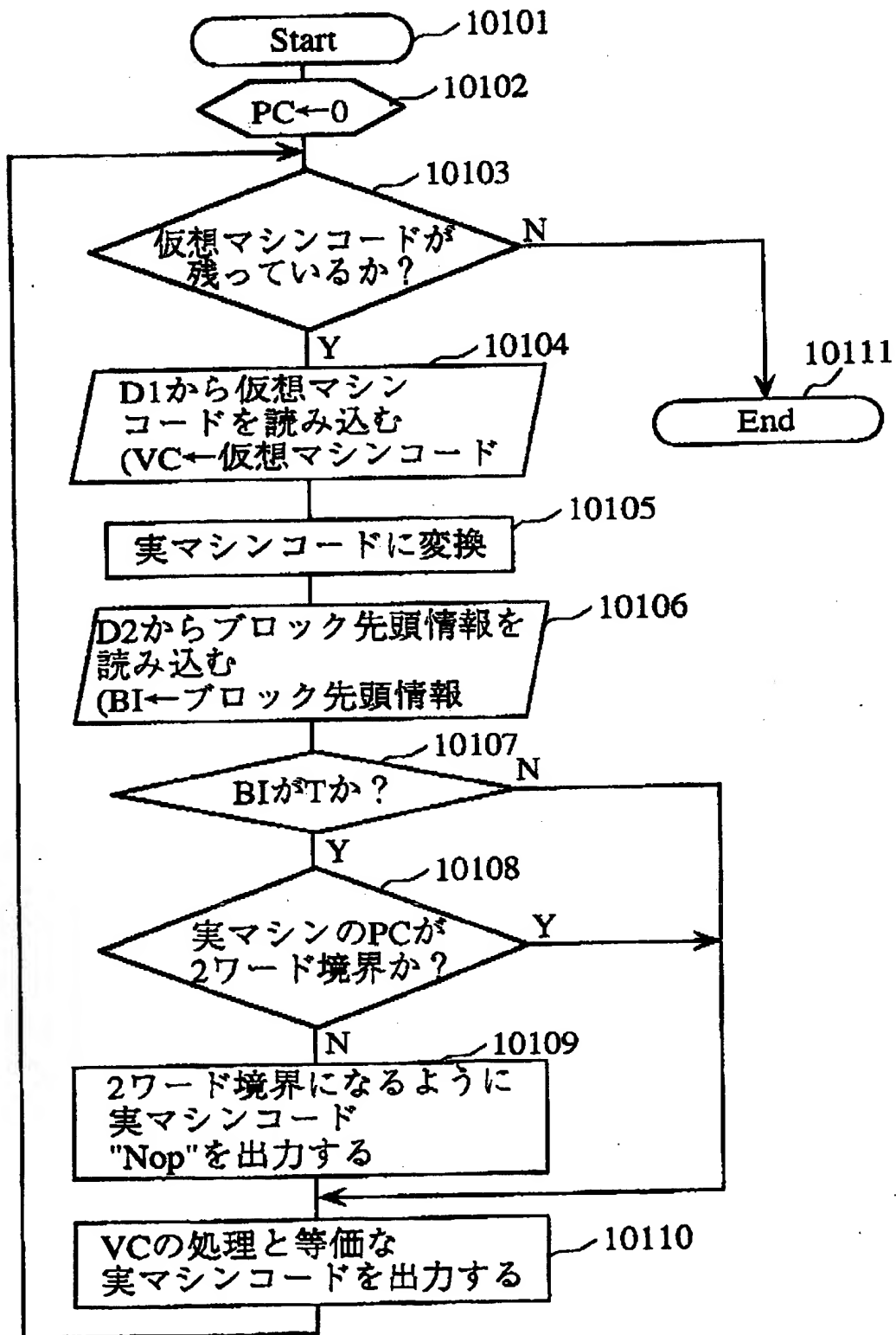
【図 66】



【図 67】



【図 68】



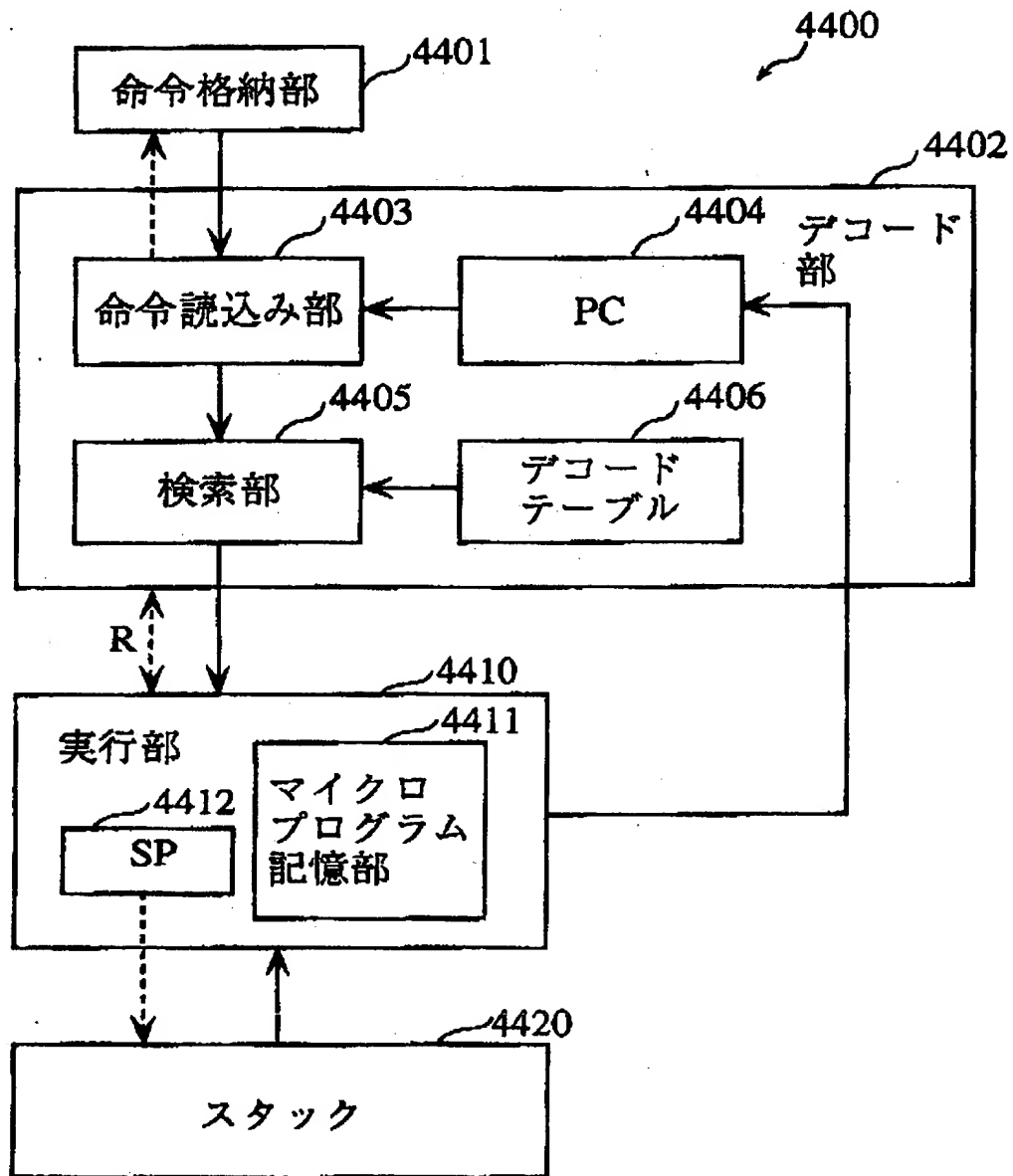
【図 69】

アドレス	仮想マシン コード	実マシンコード のサイズ	実マシンコード の該当アドレス	ブロック 先頭情報	Nop 出力
0	Push 0	4	0-3	T	-
2	Pop [0]	5	4-8	N	-
4	Push 0	4	9-12	N	-
6	Pop [1]	5	13-17	N	-
8	Push [0]	5	18-22	T	-
10	Push 10	4	23-26	N	-
12	Sub	3	27-29	N	-
13	Brz 31	5	30-34	N	Nop
15	Push [1]	5	36-40	T	-
17	Push [0]	5	41-45	N	-
19	Add	3	46-48	N	-
20	Pop [1]	5	49-53	N	-
22	Push [0]	5	54-58	N	-
24	Push 1	4	59-62	N	-
26	Add	3	63-65	N	-
27	Pop [0]	5	66-70	N	-
29	Br 8	3	71-73	N	-
31	Stop	2	74-75	T	-

【図 70】

U/D	N/T	仮想マシンオペ コード	オペランド
-----	-----	----------------	-------

【図 71】



【図 72】

仮想マシン 命令	意味	スタックの 状態	スタック ポインタの値
Push	オペランドをスタックにプッシュする	$s0 \leftarrow \text{operand}$	$sp \leftarrow sp+1$
Pop	オペランドで指定されるアドレスに、スタック トップから二番目の値を格納する	$\text{operand} \leftarrow s0$ $s0 \leftarrow s1$	$sp \leftarrow sp-1$
Add	スタックトップの値にスタックトップから二番目 の値を加え、結果をスタックトップに格納する	$s0 \leftarrow s0+s1$	$sp \leftarrow sp-1$
Mult	スタックトップの値にスタックトップから二番目 の値を掛け、結果をスタックトップに格納する	$s0 \leftarrow s0*s1$	$sp \leftarrow sp-1$
Br	オペランドで指定されるアドレスに、無条件 ジャンプする	$s0 \leftarrow s0$	$sp \leftarrow sp$
Brz	スタックトップの値が0のとき、オペランドで 指定されるアドレスにジャンプする	削除 $\leftarrow s0$ $s0 \leftarrow s1$	$sp \leftarrow sp-1$
Bmz	スタックトップの値が0以外るとき、オペランドで 指定されるアドレスにジャンプする	削除 $\leftarrow s0$ $s0 \leftarrow s1$	$sp \leftarrow sp-1$
Call	オペランドで指定されるアドレスの関数を 呼び出す	$s0 \leftarrow \text{次の仮想マシンコードアドレス}$	$sp \leftarrow sp+1$
Ret	スタックトップの値のアドレスに、無条件 ジャンプする	削除 $\leftarrow s0$ $s0 \leftarrow s1$	$sp \leftarrow sp-1$
Stop	仮想マシンの実行を停止する	初期状態	$sp \leftarrow 0$

【図 73】

オペコード	ジャンプアドレス	オペランド 数
:	:	:
Push	<Pushを処理するコードへのジャンプアドレス>	1
Pop	<Popを処理するコードへのジャンプアドレス>	1
Add	<Addを処理するコードへのジャンプアドレス>	0
Sub	<Subを処理するコードへのジャンプアドレス>	0
Inc	<Incを処理するコードへのジャンプアドレス>	0
Dec	<Decを処理するコードへのジャンプアドレス>	0
Mult	<Multを処理するコードへのジャンプアドレス>	0
Div	<Divを処理するコードへのジャンプアドレス>	0
:	:	:



【図 7 4】

(a)

仮想マシン命令"Push"のマイクロプログラム		
1:Inc	r3	: 仮想マシンのSPの値を1増やす
2:Load	r0,[r2]	: オペランドを取り出し、 レジスタ1番に読み込む
3:Inc	r2	: 仮想マシンのPCを一つ増やし、 次の命令を読み込めるようにする
4:Store	[r3],r0	: レジスタ0番の内容を、 スタックに格納する
<次の仮想マシン命令にジャンプするためのマイクロプログラム>		

(b)

仮想マシン命令"Add"のマイクロプログラム		
1:Load	r0,[r3]	: スタックから値を取り出し、 レジスタ0番に読み込む
2:Dec	r3	: 仮想マシンのSPの値を1減らす
3:Load	r1,[r3]	: スタックから値を取り出し、 レジスタ1番に読み込む
4:Add	r0,r0,r1	: レジスタ0番とレジスタ1番を加え 結果をレジスタ1番に格納する
5:Store	[r3],r0	: レジスタ0番の内容を、 スタックに格納する
<次の仮想マシン命令にジャンプするためのマイクロプログラム>		

(c)

仮想マシン命令"Mult"のマイクロプログラム		
1:Load	r0,[r3]	: スタックから値を取り出し、 レジスタ0番に読み込む
2:Dec	r3	: 仮想マシンのSPの値を1減らす
3:Load	r1,[r3]	: スタックから値を取り出し、 レジスタ1番に読み込む
4:Mult	r0,r0,r1	: レジスタ0番とレジスタ1番を掛け 結果をレジスタ1番に格納する
5:Store	[r3],r0	: レジスタ0番の内容を、 スタックに格納する
<次の仮想マシン命令にジャンプするためのマイクロプログラム>		

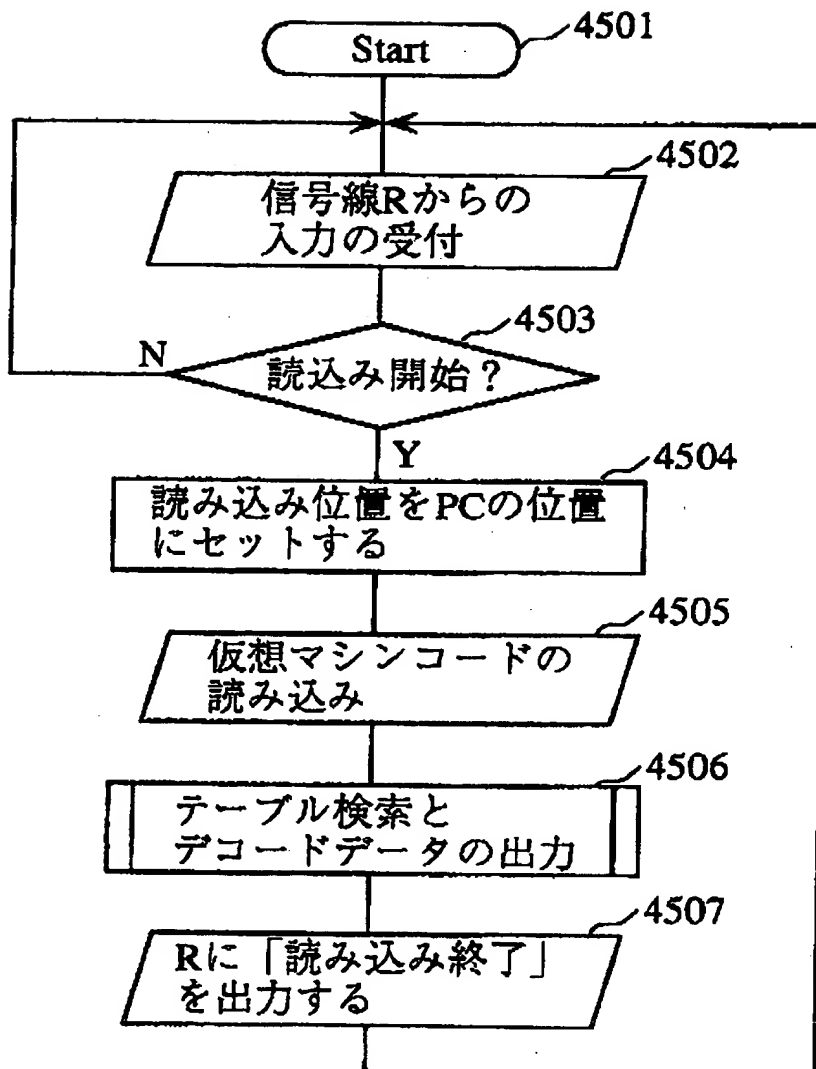
(d)

<次の仮想マシン命令にジャンプするためのマイクロプログラム>		
1:Load	r0,[r2]	: 仮想マシンのPCの位置の仮想 マシン命令(ジャンプアドレス)を レジスタ0番に読み込む
2:Inc	r2	: 仮想マシンのPCの値を1増やす
3:Jmp	r0	: レジスタ0番で示される位置に 無条件ジャンプする

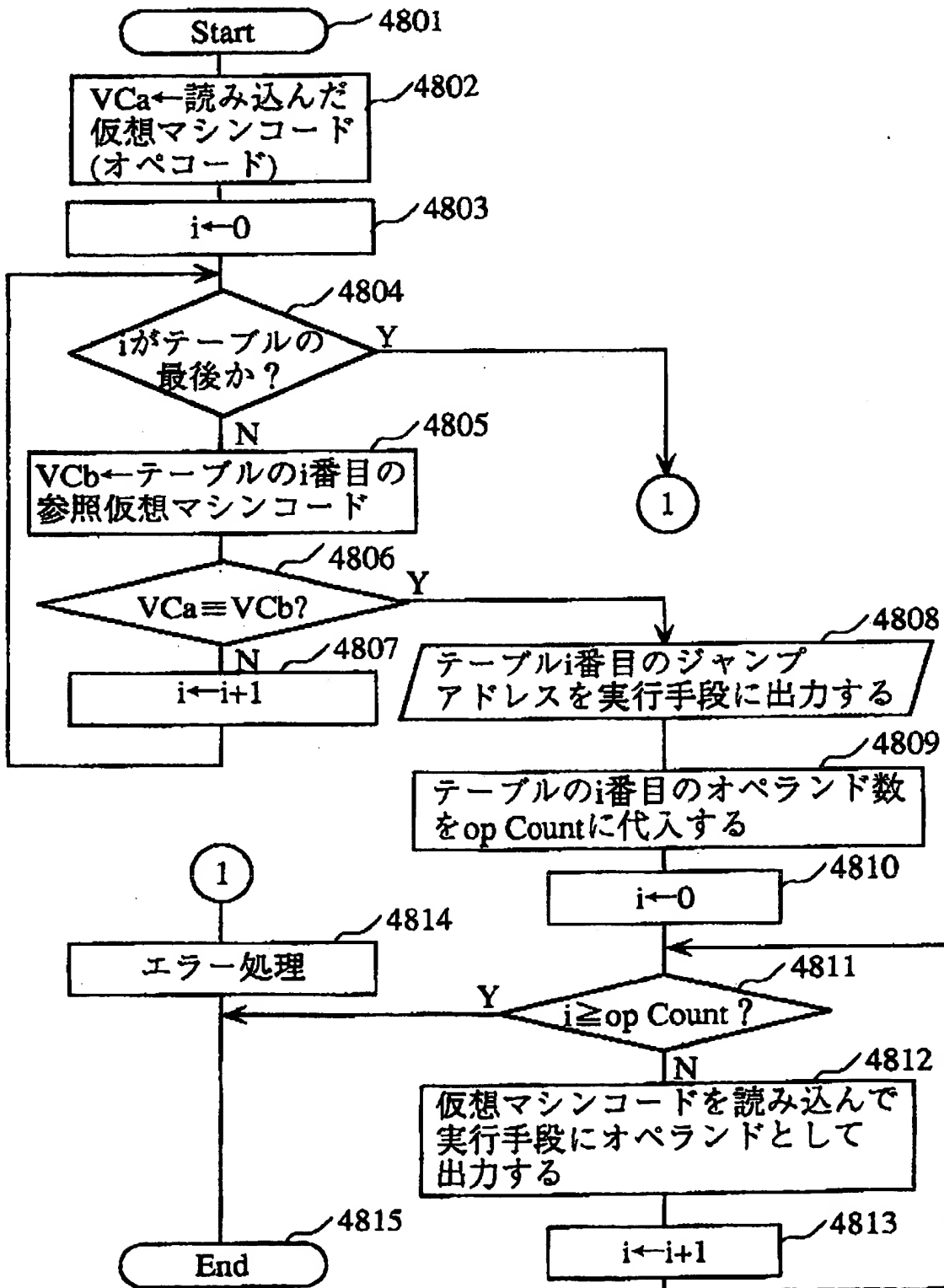
【図 75】

実マシン命令	意味	表記方法	例
Load	指定されたレジスタ(x0)の値に、指定されたレジスタの値(x1)か、レジスタで指定されたメモリ位置([x1])の値を格納する	Load x0,x1 Load x0,[x1]	Load r0,r1 Load r1,[r2]
Store	指定されたレジスタ(x1)の値を、レジスタで指定されたメモリ位置([x0])に格納する	Store [x0],x1	Store [r0],r1
Add	指定されたレジスタ(x1,x2)の値を加算し、値を指定されたレジスタ(x0)に格納する	Add x0,x1,x2	Add r0,r1,r2
Mult	指定されたレジスタ(x1,x2)の値を乗除し、値を指定されたレジスタ(x0)に格納する	Mult x0,x1,x2	Mult r0,r1,r2
Inc	指定されたレジスタ(x0)の値に1を加え、結果を同じレジスタに格納する	Inc x0	Inc r4
Dec	指定されたレジスタ(x0)の値に1を引き、結果を同じレジスタに格納する	Dec x0	Dec r3
Jump	指定されたレジスタ(x0)の値のアドレスにジャンプする	Jump x0	Jump r2
Jz	指定されたレジスタ(x0)の値が0ならば、指定されたレジスタ(x1)のアドレスにジャンプする	Jz x0,x1	Jz r0,r2
Jnz	指定されたレジスタ(x0)の値が0以外ならば、指定されたレジスタ(x1)のアドレスにジャンプする	Jnz x0,x1	Jnz r0,r2
Nop	何もしない	Nop	Nop

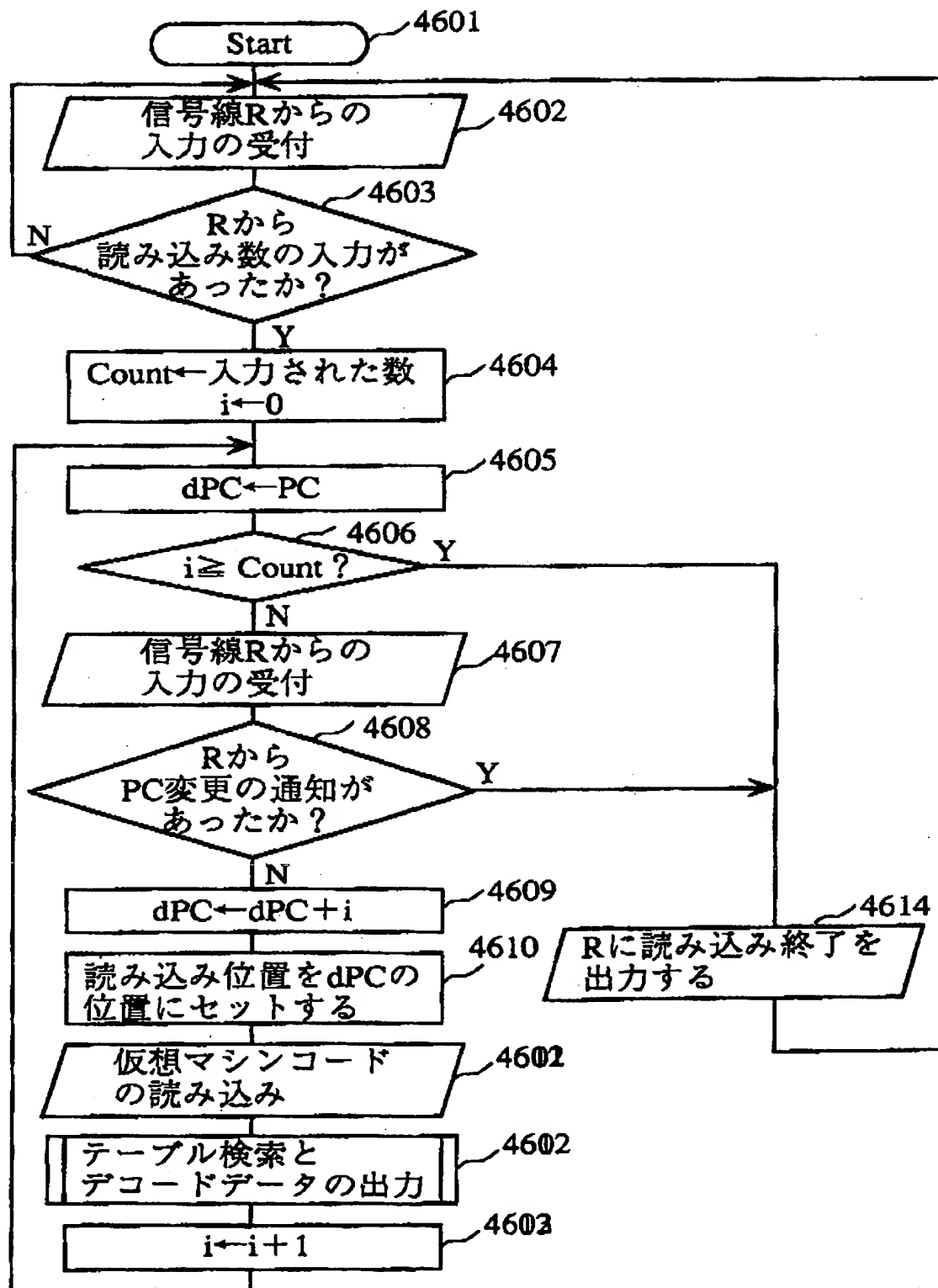
【図 76】



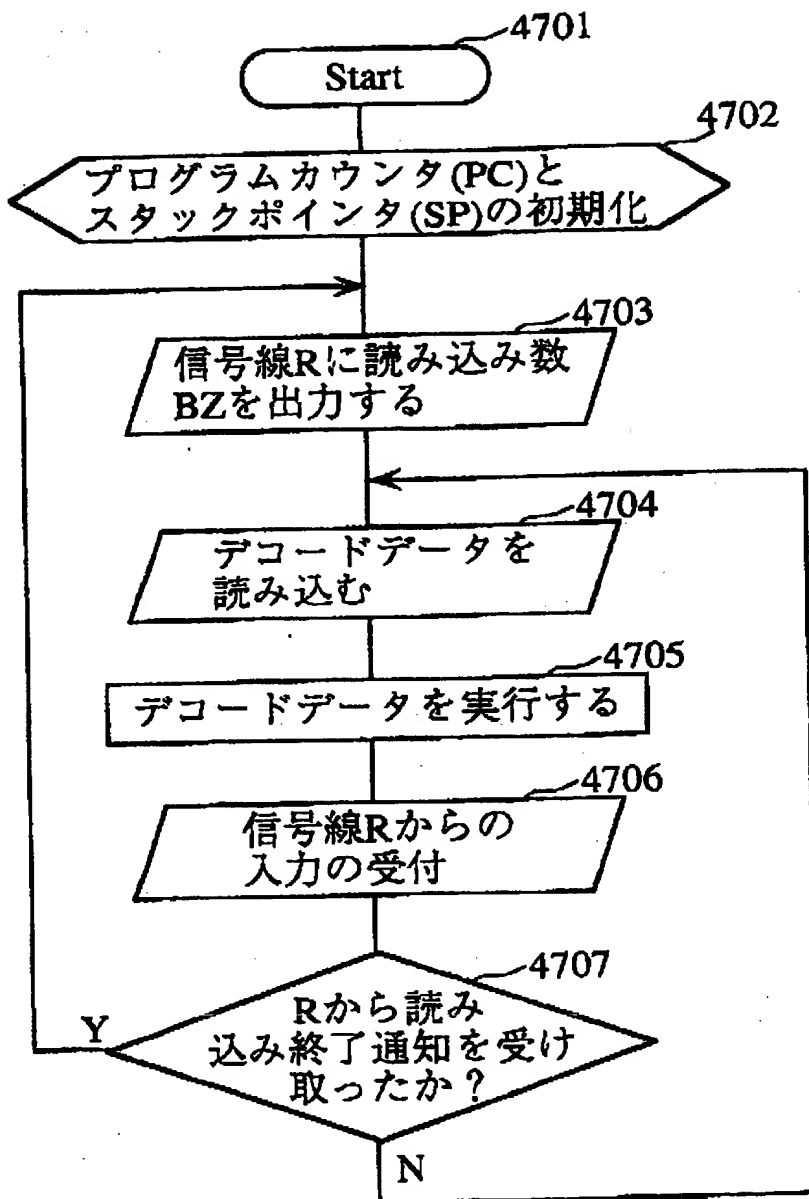
【図 77】



【図 78】



【図 79】



【図 80】

(a)

1:	Push
2:	2
3:	Push
4:	3
5:	Push
6:	4
7:	Add
8:	Mult
9:	Pop
10:	0

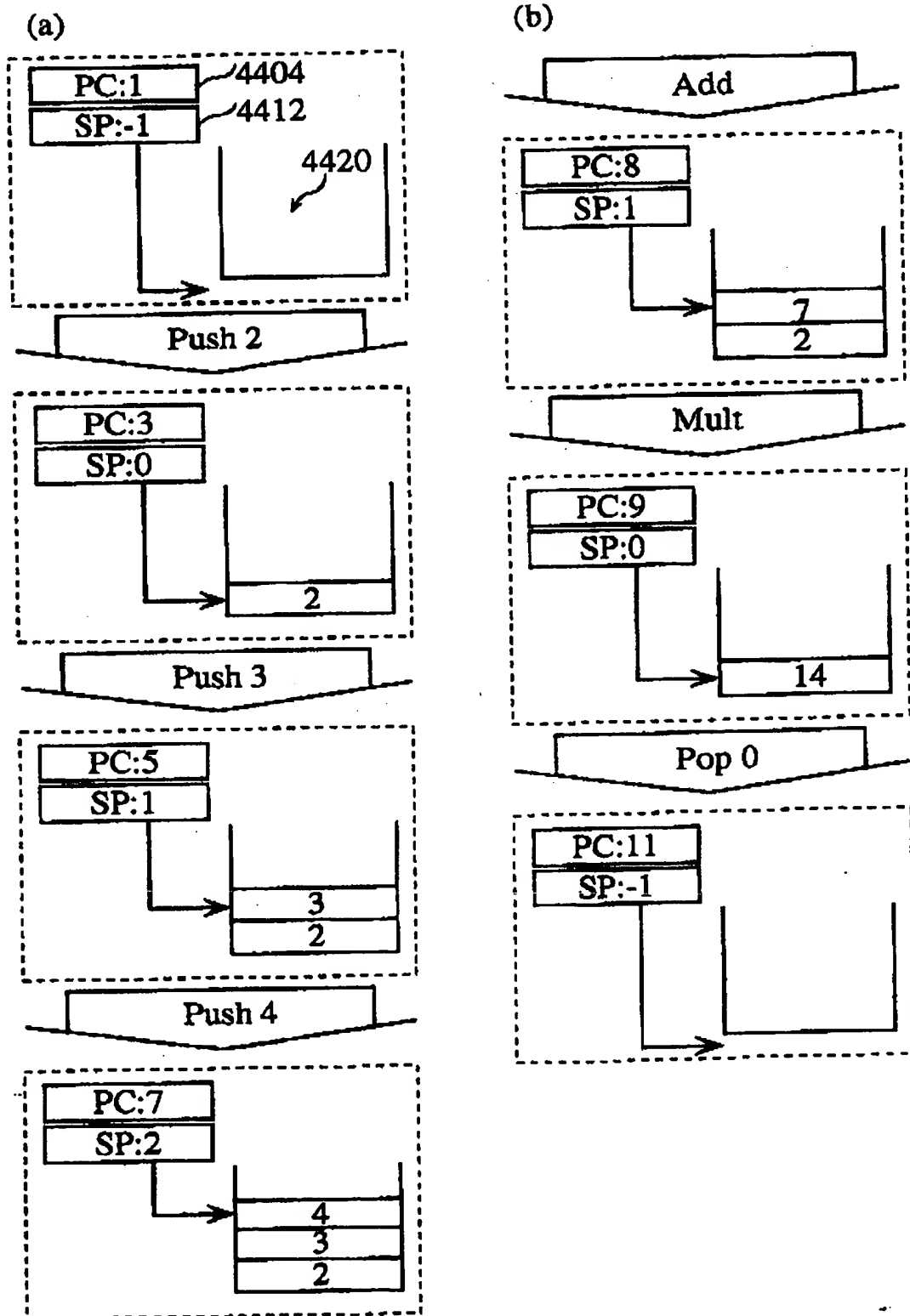
(b)

演算式：＜データ領域0番＞=2\*(3+4)

(c)

1:	＜Pushを処理するコードへのジャンプアドレス＞
2:	オペランド"2"
3:	＜Pushを処理するコードへのジャンプアドレス＞
4:	オペランド"3"
5:	＜Pushを処理するコードへのジャンプアドレス＞
6:	オペランド"4"
7:	＜Addを処理するコードへのジャンプアドレス＞
8:	＜Multを処理するコードへのジャンプアドレス＞
9:	＜Popを処理するコードへのジャンプアドレス＞
10:	オペランド"0"

【図 8 1】





【図 8 2】

(a)

仮想マシン命令"Push"のマイクロプログラム		
1:Inc	r3	; 仮想マシンのSPの値を1増やす
2:Store	[r3],r0	; TOSレジスタ(0番)の値を スタックに格納する
3:Load	r0,[r2]	; オペランドを取り出し、 TOSレジスタに読み込む
4:Inc	r2	; 仮想マシンのPCを一つ増やし、 次の命令を読み込めるようにする
<次の仮想マシン命令にジャンプするためのマイクロプログラム>		

(b)

仮想マシン命令"Add"のマイクロプログラム		
1:Load	r1,[r3]	; スタックから値を取り出し、 レジスタ1番に読み込む
2:Dec	r3	; 仮想マシンのSPの値を1減らす
3:Add	r0,r0,r1	; レジスタ0番とレジスタ1番を加え、 結果をTOSレジスタに格納する
<次の仮想マシン命令にジャンプするためのマイクロプログラム>		

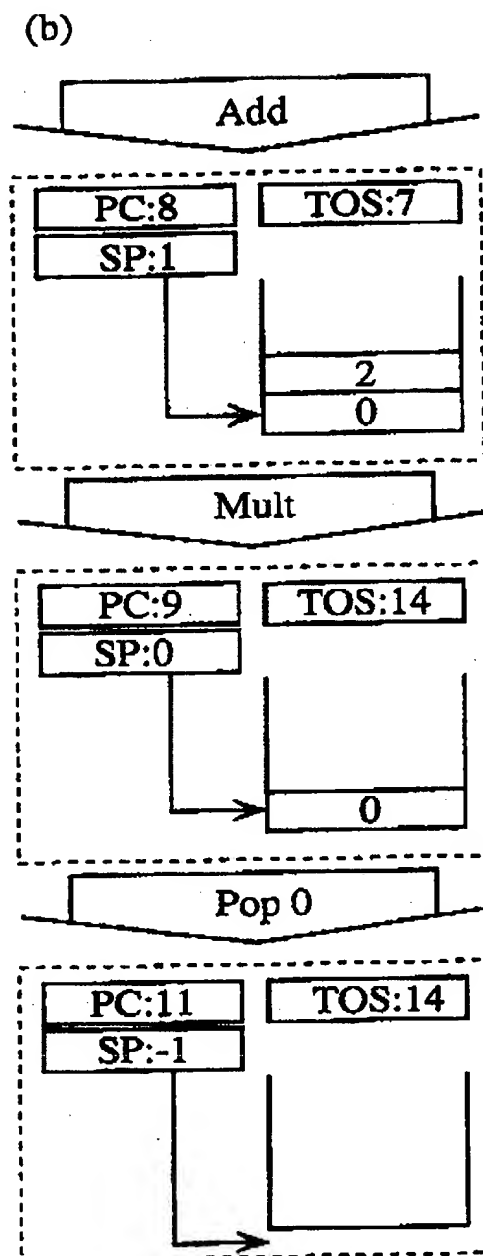
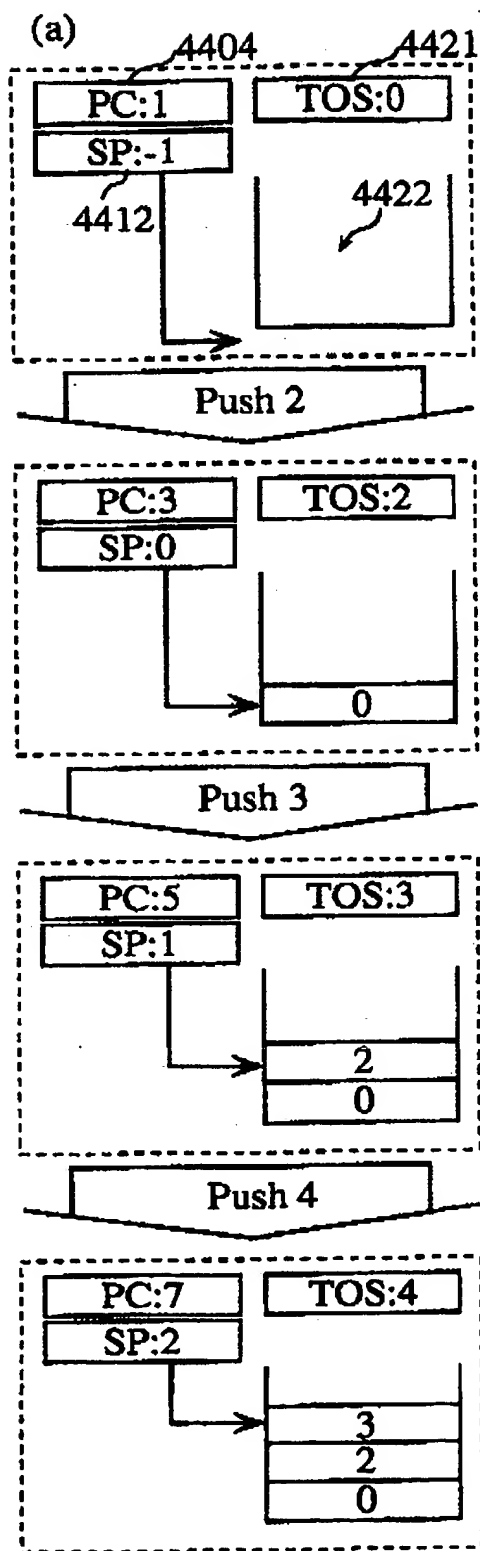
(c)

仮想マシン命令"Mult"のマイクロプログラム		
1:Load	r1,[r3]	; スタックから値を取り出し、 レジスタ1番に読み込む
2:Dec	r3	; 仮想マシンのSPの値を1減らす
3:Mult	r0,r0,r1	; レジスタ0番とレジスタ1番を掛け、 結果をTOSレジスタに格納する
<次の仮想マシン命令にジャンプするためのマイクロプログラム>		

(d)

<次の仮想マシン命令にジャンプするためのマイクロプログラム>		
1:Load	r1,[r2]	; 仮想マシンのPCの位置の仮想 マシン命令(ジャンプアドレス)を レジスタ1番に読み込む
2:Inc	r2	; 仮想マシンのPCの値を1増やす
3:Jmp	r1	; レジスタ1番で示される位置に 無条件ジャンプする

【図 8 3】



【図 84】

ステージ名	表記方法
インストラクションフェッチ デコード&レジスタ参照 実行 メモリ参照 レジスタ書き込み	IF RF ALU MEM WB

【図85】

クロック	1	2	3	4	5	6	7	8
命令A	IF	RF	ALU	MEM	WB			
命令B		IF	RF	ALU	MEM	WB		
命令C			IF	RF	ALU	MEM	WB	
命令D				IF	RF	ALU	MEM	WB

【図86】

クロック	1	2	3	4	5	6	7	8
命令A1	IF	RF	ALU	MEM	WB			
命令A2	IF	RF	ALU	MEM	WB			
命令B1		IF	RF	ALU	MEM	WB		
命令B2		IF	RF	ALU	MEM	WB		
命令C1			IF	RF	ALU	MEM	WB	
命令C2			IF	RF	ALU	MEM	WB	
命令D1				IF	RF	ALU	MEM	WB
命令D2				IF	RF	ALU	MEM	WB

【図87】

クロック	1	2	3	4	5	6	7	8	9
命令A	IF	RF	ALU	MEM	WB				
命令B		IF	RF	・	ALU	MEM	WB		
命令C			IF	・	RF	ALU	MEM	WB	
命令D				・	IF	RF	ALU	MEM	WB

【図 88】

クロック	1	2	3	4	5	6	7	8	9
命令A1	IF	RF	ALU	MEM	WB				
命令A2	IF	RF	ALU	MEM	WB				
命令B1		IF	RF	・	ALU	MEM	WB		
命令B2		IF	RF	ALU	MEM	WB			
命令C1			IF	RF	ALU	MEM	WB		
命令C2			IF	RF	・	ALU	MEM	WB	
命令D1				IF	RF	ALU	MEM	WB	
命令D2				IF	RF	・	ALU	MEM	WB

【図 89】

クロック	1	2	3	4	5	6	7	8	9	10
命令A	IF	RF	ALU	MEM	WB					
命令B		IF	RF	・	・	ALU	MEM	WB		
命令C			IF	・	・	RF	ALU	MEM	WB	
命令D				・	・	IF	RF	ALU	MEM	WB

【図 90】

クロック	1	2	3	4	5	6	7	8	9
命令A1	IF	RF	ALU	MEM	WB				
命令A2	IF	RF	ALU	MEM	WB				
命令B1		IF	RF	・	・	ALU	MEM	WB	
命令B2		IF	RF	ALU	MEM	WB			
命令C1			IF	RF	ALU	MEM	WB		
命令C2			IF	RF	ALU	MEM	WB		
命令D1				IF	RF	ALU	MEM	WB	
命令D2				IF	RF	・	ALU	MEM	WB

【図 9 1】

クロック	1	2	3	4	5	6	7
命令A	IF	RF	ALU	MEM	WB		
命令B		IF	x				
命令C			IF	RF	ALU	MEM	WB

【図 9 2】

クロック	1	2	3	4	5	6	7
命令A1	IF	RF	ALU	MEM	WB		
命令A2	IF	RF	ALU	MEM	WB		
命令B1		IF	x				
命令B2		IF	x				
命令C1			IF	RF	ALU	MEM	WB
命令C2			IF	RF	ALU	MEM	WB

【図 9 3】

クロック	1	2	3	4	5	6	7	8	9	10	11
Load r1,[r2]	IF	RF	ALU	MEM	WB						
Inc r2		IF	RF	ALU	MEM	WB					
Jmp r1			IF	RF	ALU	MEM	WB				
				IF	x						
Load r1,[r3]					IF	RF	ALU	MEM	WB		
Dec r3						IF	RF	ALU	MEM	WB	
Mult r0,r0,r1							IF	RF	ALU	MEM	WB

【図 94】

クロック	1	2	3	4	5	6	7	8	9	10	11
Load r1,[r2]	IF	RF	ALU	MEM	WB						
Inc r2	IF	RF	ALU	MEM	WB						
Jmp r1		IF	RF	•	ALU	MEM	WB				
		IF	RF	ALU	x						
			IF	RF	x						
			IF	RF	x						
				IF	x						
				IF	x						
Load r1,[r3]					IF	RF	ALU	MEM	WB		
Dec r3					IF	RF	ALU	MEM	WB		
Mult r0,r0,r1						IF	RF	•	ALU	MEM	WB

【図 95】

ク ロ ッ ク	1	2	3	4	5	6	7	8	9	10	11	12	13
Load r1,[r2]	IF	RF	ALU	MEM	WB								
Inc r2		IF	RF	ALU	MEM	WB							
Jmp r1			IF	RF	.	ALU	MEM	WB					
				IF	.	x							
Load r1,[r3]					.	IF	RF	ALU	MEM	WB			
Dec r3							IF	RF	ALU	MEM	WB		
Multi r0,r0,r1								IF	RF	.	ALU	MEM	WB



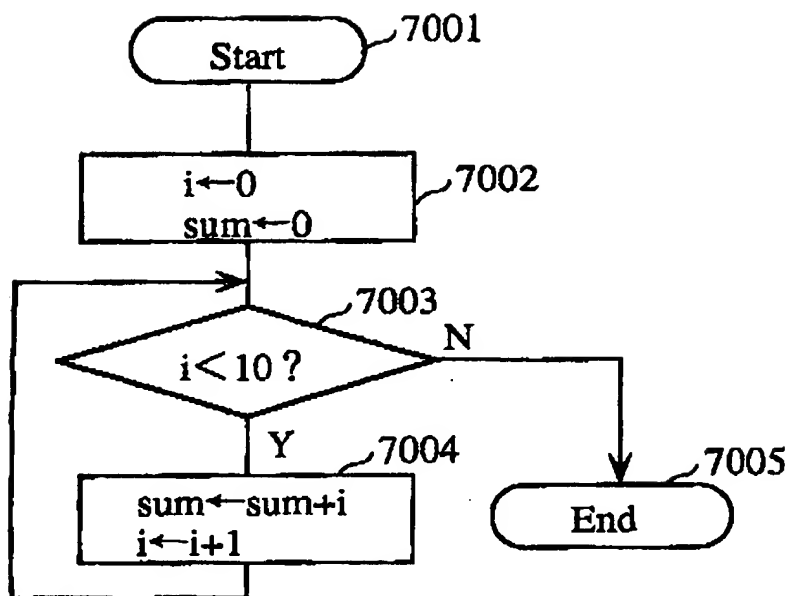
【図96】

クロック	1	2	3	4	5	6	7	8	9	10	11	12	13
Load r1,[r2]	IF	RF	ALU	MEM	WB								
Inc r2	IF	RF	ALU	MEM	WB								
Jmp r1		IF	RF	•	•	ALU	MEM	WB					
		IF	RF	ALU	MEM	x							
			IF	RF	ALU	x							
			IF	RF	ALU	x							
				IF	RF	x							
				IF	RF	x							
					IF	x							
					IF	x							
Load r1,[r3]						IF	RF	ALU	MEM	WB			
Dec r3						IF	RF	ALU	MEM	WB			
Mult r0,r0,r1							IF	RF	•	•	ALU	MEM	WB

【図 97】

0:Push	0	;i←0
2:Pop	[0]	
4:Push	0	;sum←0
6:Pop	[1]	
8:Push	[0]	;i<10?
10:Push	10	
12:Sub		
13:Brz	31	
15:Push	[1]	;sum←sum+i
17:Push	[0]	
19:Add		
20:Pop	[1]	
22:Push	[0]	;i←i+1
24:Push	1	
26:Add		
27:Pop	[0]	
29:Br	8	
31:Stop		

【図 98】



【図 99】

仮想マシン 命令	実マシンコード テンプレートサイズ	実マシンコード テンプレート	オペランド 数
：	：	：	：
Push	4ワード	<Push>を処理するコード>	1
Pop	5ワード	<Pop>を処理するコード>	1
Add	3ワード	<Add>を処理するコード>	0
Sub	3ワード	<Sub>を処理するコード>	0
Mult	3ワード	<Mult>を処理するコード>	0
Push[]	5ワード	<Push[]>を処理するコード>	1
Br	3ワード	 を処理するコード>	1
Brz	5ワード	<Brz>を処理するコード>	1
Stop	2ワード	<Stop>を処理するコード>	0
：	：	：	：

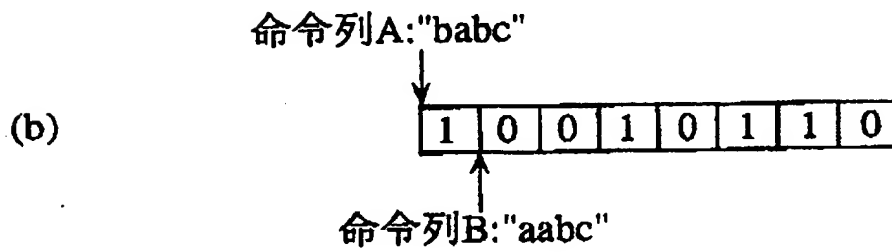
【図 100】

仮想マシン コードのアドレス	仮想マシン コード	実マシンコード のサイズ	実マシンコード の該当アドレス
0	Push 0	4	0-3
2	Pop [0]	5	4-8
4	Push 0	4	9-12
6	Pop [1]	5	13-17
8	Push [0]	5	18-22
10	Push 10	4	23-26
12	Sub	3	27-29
13	Brz 31	5	30-34
15	Push [1]	5	35-39
17	Push [0]	5	40-44
19	Add	3	45-47
20	Pop [1]	5	48-52
22	Push [0]	5	53-57
24	Push 1	4	58-61
26	Add	3	62-64
27	Pop [0]	5	65-69
29	Br 8	3	70-72
31	Stop	2	73-75

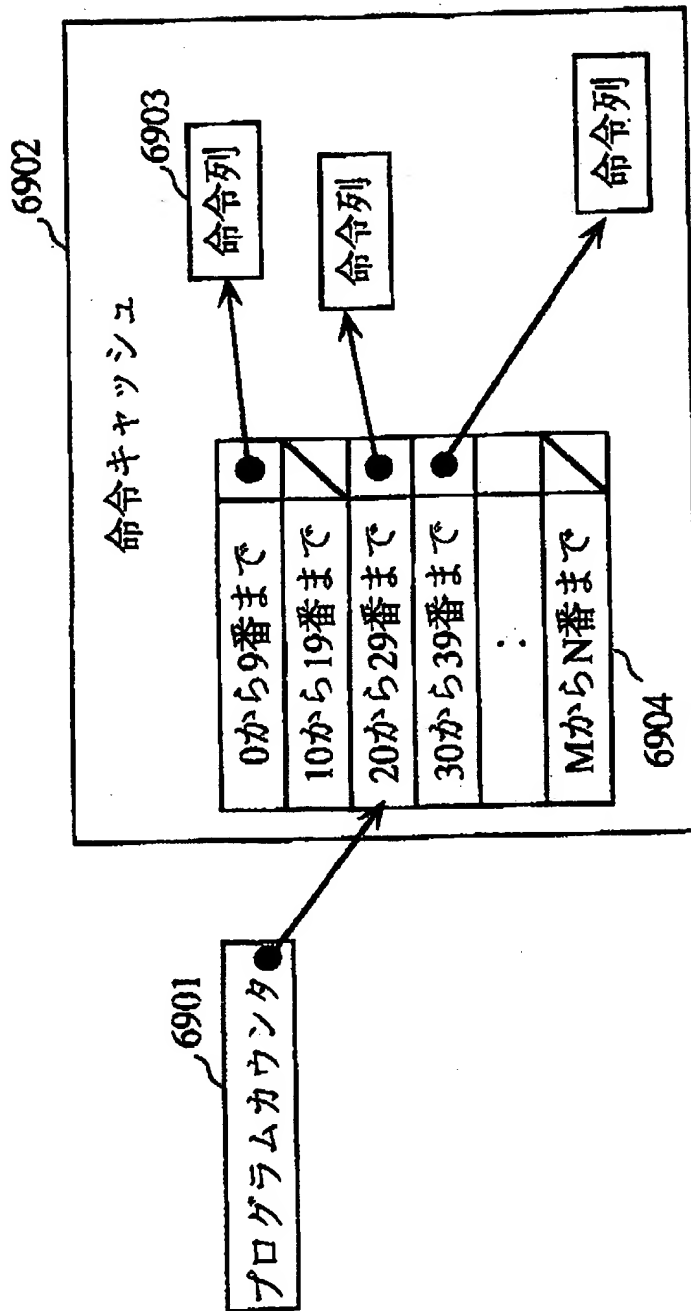
【図 101】

(a)

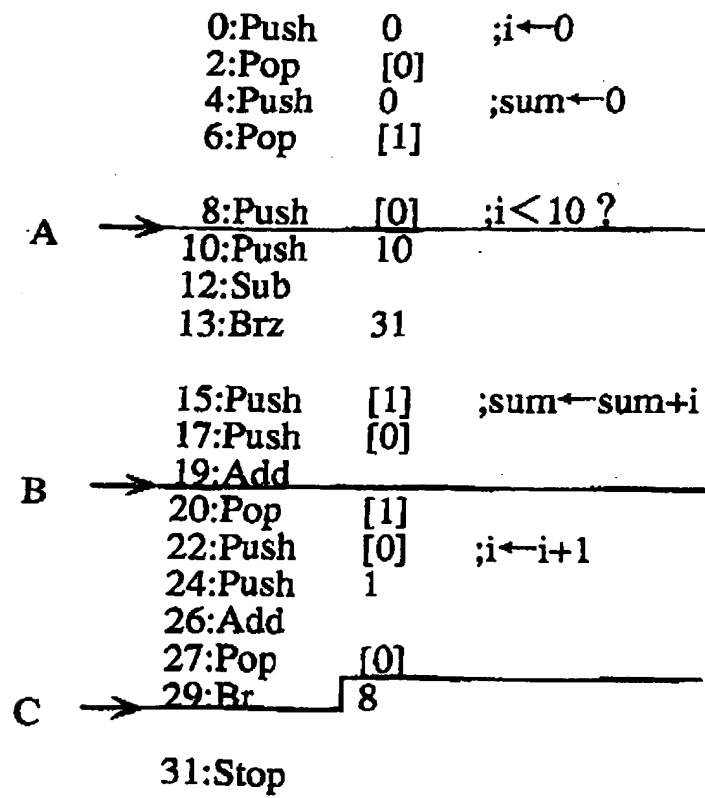
ビット列	意味
0	"a"
10	"b"
110	"c"
111	"d"



【図 102】



【図 103】





【書類名】 要約書

【要約】

【課題】 データ依存関係を吸収することにより高速実行を可能とした仮想マシンを提供する。

【解決手段】 スタック型仮想マシンであって、最上段TOSと次段SOSが実マシン201のレジスタに割り当てられたスタック120と、実行対象となる仮想マシン命令列が格納された命令格納部102と、それら仮想マシン命令それぞれに対応づけられた情報であって、対応づけられた仮想マシン命令に後続する仮想マシン命令が実行された場合のスタック120での記憶段数が増加するか減少するかを示す先行命令情報が格納された先行命令情報格納部101と、それらから次に実行すべき仮想マシン命令及び対応する先行命令情報を読み出して解読するデコード部103と、解読された仮想マシン命令の演算処理と先行命令情報に基づくスタック120の先行処理とを実行する実行部110とを備える。

【選択図】 図2

【書類名】 職権訂正データ  
【訂正書類】 特許願

<認定情報・付加情報>

【特許出願人】

【識別番号】 000005821

【住所又は居所】 大阪府門真市大字門真 1006 番地

【氏名又は名称】 松下電器産業株式会社

【代理人】 申請人

【識別番号】 100090446

【住所又は居所】 大阪市北区豊崎 3 丁目 2 番 1 号 淀川 5 番館 6 F

中島国際特許事務所

【氏名又は名称】 中島 司朗

出 願 人 履 歴 情 報

識別番号 [000005821]

1. 変更年月日 1990年 8月28日

[変更理由] 新規登録

住 所 大阪府門真市大字門真1006番地  
氏 名 松下電器産業株式会社